

**PCT**WORLD INTELLECTUAL PROPERTY ORGANIZATION  
International Bureau

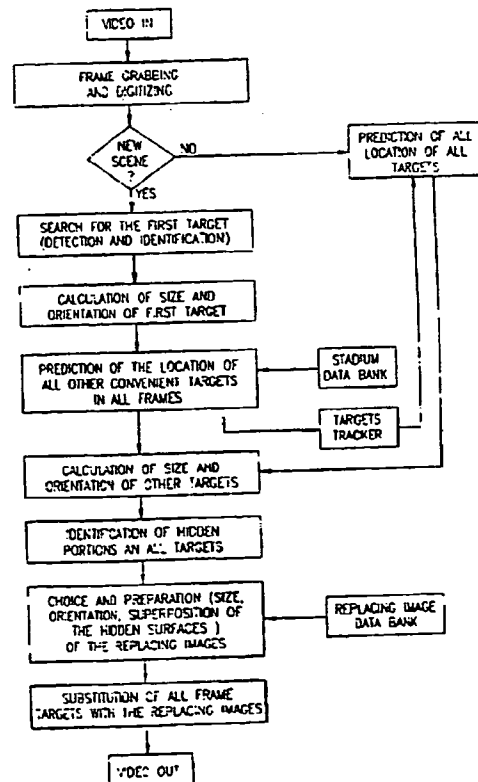
## INTERNATIONAL APPLICATION PUBLISHED UNDER THE PATENT COOPERATION TREATY (PCT)

(51) International Patent Classification <sup>5</sup> : <b>H04N 7/18</b>		<b>A1</b>	(11) International Publication Number: <b>WO 95/10919</b>
			(43) International Publication Date: 20 April 1995 (20.04.95)
(21) International Application Number: <b>PCT/US94/01679</b>		(81) Designated States: AT, AU, BB, BG, BR, BY, CA, CH, CN, CZ, DE, DK, ES, FI, GB, HU, JP, KP, KR, KZ, LK, LU, LV, MG, MN, MW, NL, NO, NZ, PL, PT, RO, RU, SD, SE, SK, UA, US, UZ, VN, European patent (AT, BE, CH, DE, DK, ES, FR, GB, GR, IE, IT, LU, MC, NL, PT, SE), OAPI patent (BF, BJ, CF, CG, CI, CM, GA, GN, ML, MR, NE, SN, TD, TG).	
(22) International Filing Date: 14 February 1994 (14.02.94)			
(30) Priority Data: 107266 12 October 1993 (12.10.93) IL 104725 14 February 1994 (14.02.94) IL			
(71) Applicant (for all designated States except US): ORAD, INC. [US/US]; Law Offices of Morse Geller, Suite 202, 116-16 Queens Boulevard, Forest Hills, NY 11375 (US).		Published With international search report.	
(72) Inventors; and (75) Inventors/Applicants (for US only): SHARIR, Avi [IL/IL]; 21 Ani Maamin Street, Ramat Hasharon 46 212 (IL). TAMIR, Michael [IL/IL]; 13 Beit Tsur Street, Ramat Aviv G, Tel Aviv 69 122 (IL).			
(74) Agents: GALLOWAY, Peter, D. et al.; Ladas & Parry, 26 West 61st Street, New York, NY 10023 (US).			

(54) Title: APPARATUS AND METHOD FOR DETECTING, IDENTIFYING AND INCORPORATING ADVERTISEMENTS IN A VIDEO

## (57) Abstract

A system (Figs 7 and 8) and method (Fig 1) for video transmission of active events, for example sports events, having in the background physical images in designated targets, wherein the physical images are electronically exchanged with preselected virtual images, so that objects or shadows actually blocking portions of the physical images will be seen by viewers as blocking the same portions of the virtual images, and the motion of players or a ball blocking the physical image will block corresponding regions of the exchanged virtual image, so that the exchanged electronic image will remain in the background of the event, exactly as the original image.



**FOR THE PURPOSES OF INFORMATION ONLY**

Codes used to identify States party to the PCT on the front pages of pamphlets publishing international applications under the PCT.

AT	Austria	GB	United Kingdom	MR	Mauritania
AU	Australia	GE	Georgia	MW	Malawi
BB	Barbados	GN	Guinea	NE	Niger
BE	Belgium	GR	Greece	NL	Netherlands
BF	Burkina Faso	HU	Hungary	NO	Norway
BG	Bulgaria	IE	Ireland	NZ	New Zealand
BJ	Benin	IT	Italy	PL	Poland
BR	Brazil	JP	Japan	PT	Portugal
BY	Belarus	KE	Kenya	RO	Romania
CA	Canada	KG	Kyrgyzstan	RU	Russian Federation
CF	Central African Republic	KP	Democratic People's Republic of Korea	SD	Sudan
CG	Congo	KR	Republic of Korea	SE	Sweden
CH	Switzerland	KZ	Kazakhstan	SI	Slovenia
CI	Côte d'Ivoire	LI	Liechtenstein	SK	Slovakia
CM	Cameroon	LK	Sri Lanka	SN	Senegal
CN	China	LU	Luxembourg	TD	Chad
CS	Czechoslovakia	LV	Latvia	TG	Togo
CZ	Czech Republic	MC	Monaco	TJ	Tajikistan
DE	Germany	MD	Republic of Moldova	TT	Trinidad and Tobago
DK	Denmark	MG	Madagascar	UA	Ukraine
ES	Spain	ML	Mali	US	United States of America
FI	Finland	MN	Mongolia	UZ	Uzbekistan
FR	France			VN	Viet Nam
GA	Gabon				

1  
2 APPARATUS AND METHOD FOR DETECTING,  
3 IDENTIFYING AND INCORPORATING ADVERTISEMENTS  
4 IN A VIDEO  
5

6 The present invention relates to apparatus  
7 and methods for superimposing a small video image into  
8 a larger video image.  
9

10  
11  
12 International sports events or other  
13 spectacles generally draw the interest and attention of  
14 spectators in many countries. For example, the  
15 Olympics, Superbowl, World Cup, major basketball and  
16 soccer games, auto races etc. fit into this category.  
17 Such events are generally broadcast live by video to a  
18 large international audience. The locale in which  
19 these events take place, such as stadiums or courts,  
20 provide advertising space all around in the form of  
21 signs, posters or other displays on fences and  
22 billboards, and in fact on any unoccupied space  
23 suitably located, including sections of the playing  
24 field.

25 Due to the nature of the displays, which are  
26 mostly in the form of printed matter, they are not  
27 changed too frequently and remain at least for a day,  
28 or a series or a whole season, and are directed mostly  
29 at local audiences. In cases where two teams from  
30 different countries play each other, the advertisements  
31 are usually arranged so that one side of the stadium  
32 contains advertisements directed to audiences in one  
33 country, while the other side has advertisements  
34 directed to the spectators in the other country.

35 The video cameras in these instances film the  
36 event from opposite sides of the stadium for their  
37 respective audiences. This of course is logistically  
38 complicated and limits the angle from which the events

1 can be seen in either of the countries represented in  
2 the game.

3 Another limitation to present methods of  
4 advertising is the stringent safety requirements for  
5 positioning the billboards, so as not to interfere with  
6 the game, nor disturb the view of the spectators in the  
7 stadium, nor pose a danger to the players. The  
8 displays must not be too close to the actual field of  
9 action, so as not to distract the players.

10 A most serious drawback of the present system  
11 for advertising at major world sports events is the  
12 fact that although the event is televised live  
13 throughout the world, the actual physical  
14 advertisements in the stadium, because of their broad  
15 international exposure, can only cater to products  
16 having a world market.

17 Local advertisers can only make use of such  
18 world-class televised events by locally superimposing  
19 messages on the TV screen, or by interrupting the real  
20 time of the event.

21 Another drawback of the existing system is  
22 that over long time periods, due to the scanning of the  
23 TV camera, the signs appear too blurred to be read by  
24 the TV viewers. On many other occasions, only part of  
25 the sign is visible to the TV viewers and the sign  
26 cannot be read.

27 The following reference, the disclosure of  
28 which is incorporated herein by reference, describes  
29 Gaussian edge detection:

30 J.F. Canny, "A computational approach to edge  
31 detection", IEEE Trans. Pattern Analysis and Machine  
32 Intelligence, Vol. 8, pp. 679-698, November, 1986.

33

34

35

36

37

38



1  
2  
3  
4           The present invention relates to a system and  
5 method for detecting, identifying and scaling in a  
6 video frame, suitable distinct targets and areas and  
7 inserting into these areas virtual images stored in the  
8 memory of the system, so that all objects or shadows in  
9 front of the distinct areas blocking portions thereof  
10 from view will be seen in a video transmission as being  
11 in front of and blocking the same portions of the areas  
12 containing virtual images.

13           A particular feature of the invention is to  
14 operate the system in real time. The invention also  
15 provides apparatus for operating the system. The  
16 invention is particularly useful for advertising in  
17 sports courts.

18           It is an object of the present invention to  
19 provide a system and method for video transmission of  
20 active events, for example sports events, having in the  
21 background physical images in designated targets,  
22 wherein the physical images are electronically  
23 exchanged with preselected virtual images, so that  
24 objects or shadows actually blocking portions of the  
25 physical images will be seen by viewers as blocking the  
26 same portions of the virtual images, and the motion of  
27 players or a ball blocking the physical image will  
28 block corresponding regions of the exchanged virtual  
29 image, so that the exchanged electronic image will  
30 remain in the background of the event, exactly as the  
31 original image.

32           In a preferred embodiment of the present  
33 invention, the physical image to be substituted is  
34 detected, recognized, and located automatically and is  
35 replaced within one TV frame so that the original image  
36 is not perceptible to the TV viewers. In this  
37 embodiment no man is required in the loop during line  
38 broadcasting.

1           Since the same physical image may be captured  
2 by a plurality of TV cameras deployed in various  
3 locations around the court, and each camera usually has  
4 a continuous zoom lens, the system is able to detect  
5 and identify a certain physical target in all possible  
6 spatial orientations and magnifications of the target.

7           The system is also capable of unequivocally  
8 identifying the scale and perspective of the physical  
9 target and normalizing the implanted virtual image into  
10 the same perspective.

11           Another object of the invention is to provide  
12 a system and method of implanting in video  
13 transmission, virtual images in predetermined "free"  
14 background areas generally unsuitable for displaying  
15 physical signs, like the sports court itself.

16           In a preferred embodiment of the present  
17 invention, the task of detection and identification of  
18 these free areas is executed automatically.

19           A further object of the present invention is  
20 to automatically identify cases in which the physical  
21 billboard appears blurred due to camera scanning or  
22 jitter and to replace the blurred sign with a clearer  
23 one or to alternatively apply the same blurring degree  
24 to the replacing sign so that it will have an  
25 appearance similar to its neighboring signs.

26           Yet another object of the present invention  
27 is to automatically identify a case in which only a  
28 small portion of the billboard is visible in the  
29 camera's field of view and to replace this small  
30 portion with the whole image of the original billboard.

31           Still another object of the invention is to  
32 automatically identify cases in which the resolution of  
33 the captured billboard image is not sufficient for the  
34 TV viewers and to electronically replace them with  
35 larger virtual billboards so that their message may be  
36 conveniently captured by the viewers.

37           Another object of the invention is to perform  
38 the implantation described above on a succession of

1 video frames.

2 Yet another object of the invention is to  
3 provide the above system and method for electronic  
4 exchange or planting of virtual images in real time.

5 A further object of the invention is to  
6 provide a system and method for video broadcasting the  
7 same event to different populations of viewers in real  
8 time, with different electronic messages substituted in  
9 the spaces occupied by physical displays.

10 Still another object of the invention is to  
11 provide a system and method for utilization of  
12 available space in a stadium unused by physical  
13 displays for the purpose of advertising by planting  
14 therein electronic virtual images during real time  
15 broadcasting of an event taking place in a stadium.

16 Still a further object of the invention is to  
17 provide apparatus for use in video transmission for  
18 exchanging physical images with virtual images or  
19 planting virtual images in unused background areas  
20 during an event in real time video transmission,  
21 without disturbing the actual transmission of the  
22 event.

23 In accordance with a preferred embodiment of  
24 the present invention, there is provided a system and  
25 method for broadcasting active events being captured by  
26 a TV camera, wherein virtual images are electronically  
27 substituted in or superimposed on targets selected from  
28 physical displays and preselected background regions,  
29 including an electronic data bank of event locales and  
30 targets therein, a memory unit for storing digitized  
31 virtual images for substitution in the targets,  
32 apparatus for grabbing and digitizing video frames,  
33 apparatus for automatic target searching in digitized  
34 video frames and for detecting targets therein,  
35 apparatus for localization, verifying and identifying  
36 the targets, apparatus for comparing the detected  
37 targets with corresponding targets in the data bank,  
38 apparatus for scaling and identifying the perspective

1 of the original target and transforming the virtual  
2 substitute image into the same scale and perspective,  
3 apparatus for real-time video tracking of a detected  
4 target throughout a succession of frames, and for the  
5 identification of target magnification (zoom) or  
6 changes in perspective, apparatus for distinguishing  
7 between non-background objects and shadows that block  
8 portions of the detected targets, apparatus for  
9 electronically transferring the objects and shadows  
10 from the original video frame to the substituted frame,  
11 apparatus for inserting the electronically transformed  
12 virtual image into the video frame substituting the  
13 original image in the target without this  
14 transformation being perceptible by the viewers,  
15 apparatus for receiving and storing virtual images and  
16 generating a virtual images data bank, apparatus for  
17 generating a locale data bank either prior or during an  
18 event (a learning system) and video signal input-output  
19 apparatus.

20 For this purpose the system uses a special  
21 method for the automatic detection and identification  
22 of targets using one or more of the following  
23 attributes:

- 24 - geometry - such as the physical configuration  
25 of billboards (rectangular shape or parallel lines  
26 attribute) as seen from different angles and  
27 magnifications,
- 28 - texture of slogans and graphics - such as for  
29 example in posters,
- 30 - character recognition,
- 31 - field or court lines - which serve as  
32 references for designating free court areas,
- 33 - standard objects that have typical shape and  
34 texture - such as post, backboard, basket and/or a  
35 player's shirt,
- 36 - colour, and
- 37 - objects and shadows temporarily blocking  
38 portions of the image intended to be exchanged.

1           The method clearly identifies the subject  
2 target at any capturing angle and range and in any zoom  
3 state, and preferably in real time, so that the  
4 original billboard will not be perceptible to the TV  
5 viewers. The method typically identifies, in any  
6 frame, a relatively large number of targets (up to 20  
7 targets or more in an extreme case).

8           Blocking objects and shadows are  
9 distinguished from the background image by means of:  
10        comparing the detected target (partially blocked)  
11 with the same target stored in the system's data bank.  
12 The smooth and processed difference image between the  
13 two is the image of hidden surfaces which forms a part  
14 of the blocking object. This procedure may be  
15 implemented also by using correlation windows and  
16 identifying a low value of the correlation coefficient  
17 as being due to occlusion,

18        motion detection - to identify objects that move  
19 with respect to the background,

20        texture and geometric shape - distinguishing a  
21 player, ball or shadow from a sign, slogan or graphic  
22 image etc., and

23        colour - and shades of colour.

24        The electronic exchange is preferably instant  
25 and unnoticeable by the viewer since a perceptible  
26 exchange is usually unaccepted by the TV networks.  
27 Alternatively, it is possible to continuously "fade"  
28 the original image while enhancing the virtual image.

29        False identification of targets and images is  
30 preferably avoided.

31        The substituted target should be localized to  
32 sub-pixel accuracy so that the replacing target be  
33 spatially fixed with respect to the frame during the  
34 whole succession of TV frames in which the target is  
35 inside the camera's field of view. This accuracy is due  
36 to the fact that the human eye is sensitive to sub-  
37 pixel motions.

38

1           The methods preferably employ special  
2 parallel and pipelined processing hardware which will  
3 allow carrying out simultaneously the large number of  
4 operations involved in this process.

5           The method of this invention preferably uses  
6 two optional sub-systems:

7 a) Digital Image Converter and Storage Unit -  
8 consisting of an electro-optical scanner for digital  
9 conversion and storage of virtual images, for  
10 constructing a memory unit for images such as  
11 advertisements. The system may also have the  
12 possibility of inputting images such as advertisements  
13 in other ways, as by digital interface (magnetic,  
14 optical disc, communication link) or video port, and  
15 may further include a graphics programme and man-  
16 machine interface for designing virtual images (like  
17 slogans) "on-the-spot".

18 b) Locale "learning" and storage system - for  
19 creating a data bank of targets and fixed objects in  
20 locales such as stadiums and fields, including: signs  
21 (location, shape, colour and type - one-time, seasonal,  
22 etc.), court markers (lines, colour, goal/basket,  
23 post), etc.

24           These two sub-systems can operate off-line or  
25 can be part of the basic system. The system can  
26 "learn" the details of the court in the course of a  
27 live event and create/update its data bank for future  
28 use. This can also be done using the trial shots taken  
29 before the event starts.

30           The method involves the following steps:

31           When the live or previously recorded video  
32 film is being transmitted, the following steps take  
33 place:

34           1) Frame grabbing and digitization - each  
35 video frame is grabbed and each pixel value is  
36 digitized and stored in system memory,

37           2) Searching - the captured video frame  
38 is scanned to detect either actual physical displays

1 (like the icons stored in the memory) or background  
2 regions suitable for implantation whose specifications  
3 have been pre-defined. After detection, suspected  
4 targets, i.e. displays, are checked for unequivocal  
5 identification. This is accomplished by identification  
6 of messages and graphics in the displays, or of colour  
7 and texture attributes using standard pattern  
8 recognition techniques like edge correlation and region  
9 matching methods, character recognition, neural  
10 network techniques and so on. After the target  
11 (display) has been identified and accurately localized,  
12 its optical magnification and perspective are computed  
13 and the locations of all other stored targets  
14 (displays) in the frame are consecutively predicted  
15 using the locale's lay-out in the data bank, giving the  
16 system positive search clues for additional targets in  
17 the same video frame.

18           3) Blocked surface identification - when a  
19 given message area or display region is positively  
20 identified in a frame, the target (display) is compared  
21 with its properly scaled stored image (icon) and those  
22 areas of the display that are temporarily blocked by an  
23 object such as by the body of a player, by a ball or a  
24 shadow etc. are revealed after proper smoothing and  
25 processing of the results. The pixel addresses of these  
26 surfaces are stored so that these surfaces will later  
27 be superimposed on the substituted image.

28           4) Scaling, perspective transformation and  
29 substitution - when a physical image display or a  
30 free location is identified and localized, the memory  
31 of the system is searched to find the desired virtual  
32 image to be substituted or implanted. The exchanged  
33 virtual image (patch) is then first normalized to  
34 acquire the proper size and perspective of the original  
35 physical image and identified blocked surfaces are then  
36 removed, so that the exchanged image looks like a  
37 background display or as a painted display on the  
38 court.

1           5) Real-time video tracking - typically a  
2 given display is visible for a few second before it  
3 moves out of the camera's field of view. The system  
4 preferably uses previous frames' information to track a  
5 given display throughout this succession of frames. To  
6 do that, conventional video tracking techniques, such  
7 as edge, centroid or correlation tracking methods, are  
8 executed. These methods should incorporate subpixel  
9 accuracy estimates. Tracking of players or of the ball  
10 can also be instrumental to identify blocking portions  
11 in the case where target icons are not stored in the  
12 system memory or for implantation in free regions.

13           There is thus provided, in accordance with a  
14 preferred embodiment of the present invention,  
15 apparatus for advertisement incorporation including a  
16 field grabber operative to grab and digitize at least  
17 one field representing at least a portion of a sports  
18 facility, and an advertisement incorporator operative  
19 to incorporate, into at least one field, an  
20 advertisement whose contents varies over time.

21           Further in accordance with a preferred  
22 embodiment of the present invention, the advertisement  
23 incorporator includes an advertisement site detector  
24 operative to detect at least one advertisement site in  
25 at least one field on a basis other than location of  
26 the advertisement site relative to the sports facility.

27           Still further in accordance with a preferred  
28 embodiment of the present invention, the advertisement  
29 incorporator is operative to incorporate an  
30 advertisement into at least one field at a partially  
31 occluded advertisement site within the sports facility.

32           Still further in accordance with a preferred  
33 embodiment of the present invention, the contents of  
34 the advertisement varies in accordance with a  
35 predetermined schedule.

36           Additionally in accordance with a preferred  
37 embodiment of the present invention, the contents of  
38 the advertisement varies in accordance with an external



1 input.

2 Further in accordance with a preferred  
3 embodiment of the present invention, the advertisement  
4 incorporator also includes an audience noise evaluator  
5 operative to detect and evaluate a level of noise  
6 generated by an audience and to provide a noise level  
7 input to the advertisement incorporator and wherein the  
8 contents of the advertisement varies in accordance with  
9 the noise level input.

10 There is additionally provided, in accordance  
11 with a preferred embodiment of the present invention,  
12 a method for advertisement incorporation including  
13 grabbing and digitizing at least one field representing  
14 at least a portion of a sports facility, and  
15 incorporating into at least one field, an advertisement  
16 whose contents varies over time.

17

18

19

20

21

22

23

24

25

26

27

28

29

30

31

32

33

34

35

36

37

38

1  
2  
3           The present invention will be understood and  
4 appreciated more fully from the following detailed  
5 description, taken in conjunction with the drawings and  
6 appendices in which:

7  
8           Fig. 1 is a logical flow diagram of the  
9 processes and tasks required in accordance with a  
10 preferred embodiment of the method of the present  
11 invention;

12           Fig. 2 is a block diagram of the basic and  
13 sub-system modules in accordance with a preferred  
14 embodiment of the present invention;

15           Fig. 3 is a block diagram of a basic  
16 processing unit;

17           Fig. 4 illustrates a minimum basic on-line  
18 system in accordance with a preferred embodiment of the  
19 present invention;

20           Fig. 5 illustrates a minimum basic off-line  
21 system in accordance with the invention;

22           Fig. 6 illustrates a system in accordance  
23 with a preferred embodiment of the present invention  
24 adapted for cable TV application;

25           Fig. 7 is a simplified block diagram of a  
26 real time system for advertisement site detection and  
27 advertisement incorporation, constructed and operative  
28 in accordance with a preferred embodiment of the  
29 present invention;

30           Fig. 8 is a simplified block diagram of the  
31 parallel processor and controller of Fig. 7;

32           Fig. 9 is a simplified block diagram of an  
33 alternative embodiment of a real time system for  
34 advertisement site detection and advertisement  
35 incorporation;

36           Fig. 10A is a simplified flowchart of a  
37 preferred method of operation of the parallel processor  
38 and controller of Fig. 7, when only a single

1 advertisement site is to be identified and only a  
2 single advertisement is to be incorporated at that  
3 site;

4           Fig. 10B is a simplified flowchart of a  
5 preferred method of operation of the parallel processor  
6 and controller of Fig. 7, when a plurality of  
7 advertisement sites is to be identified and a  
8 corresponding plurality of advertisements, which may or  
9 may not differ in content, is to be incorporated at  
10 those sites;

11           Fig. 11 is a simplified flowchart of a  
12 preferred method for performing the segmentation step  
13 of Figs. 10A and 10B;

14           Fig. 12 is a simplified flowchart of a  
15 preferred model matching method for performing the  
16 advertisement content identification step of Figs. 10A  
17 and 10B;

18           Fig. 13 is a simplified flowchart of a  
19 preferred method for performing the localization step  
20 of Figs. 10A and 10B;

21           Fig. 14 is a simplified flowchart of a  
22 preferred method for performing the tracking step of  
23 Figs. 10A and 10B;

24           Fig. 15 is a simplified flowchart of a  
25 preferred method for performing the occlusion analysis  
26 step of Figs. 10A and 10B;

27           Fig. 16 is a simplified flowchart of a  
28 preferred method for performing the advertisement  
29 incorporation step of Figs. 10A and 10B;

30           Fig. 17 is a simplified block diagram of  
31 camera monitoring apparatus useful in conjunction with  
32 the advertisement site detection/incorporation  
33 apparatus of Fig. 7;

34           Fig. 18 is a simplified flowchart of a  
35 preferred method for processing the output of the  
36 occlusion analysis process of Fig. 15 in order to take  
37 into account images from at least one off-air camera;

38           Fig. 19 is a simplified flowchart of a

1 preferred method for detecting and tracking moving  
2 objects of central interest; and  
3       Appendix A is a computer listing of a  
4 software implemented non-real time system for  
5 advertisement site detection and advertisement  
6 incorporation, constructed and operative in accordance  
7 with an alternative embodiment of the present  
8 invention.

9  
10  
11  
12  
13  
14  
15  
16  
17  
18  
19  
20  
21  
22  
23  
24  
25  
26  
27  
28  
29  
30  
31  
32  
33  
34  
35  
36  
37  
38

1

2

3

4

5

6

7

8

9

Referring now to Fig. 1, in a preferred embodiment of the present invention, the system and method are designed to automatically perform the substitution of physical targets with synthetic images in real time, although a simpler version of the invention can be used off-line.

10

11

12

When operating the system, the modules required are illustrated in the block diagram of Fig. 2. These include:

13

14

15

16

17

18

19

20

21

22

23

24

25

26

27

28

29

30

31

32

33

34

35

36

37

38

- a basic processing unit;
- an optional scanner/digitizer used to create the data bank of synthetic images from still pictures; and
- an optional sub-system composed of a TV camera, digitizer and memory to create the stadium data bank.

As was mentioned before, there may be other methods to create the data bank of synthetic images. The locale's (stadium's) data bank may also be created from the trial shots taken before the game starts or even be incrementally built in the course of the game by means of a "learning" process or by using data supplied by the stadium owner, the advertiser or the TV network.

Fig. 2 illustrates a block diagram of the apparatus used in the system, wherein 1, 2, ....n are a plurality of TV cameras in different positions, which are the usual TV network cameras, 3 is the basic processing unit described in Fig. 3, sub-system 4 converts and stores synthetic images and sub-system 5 is a "learning" and storage system for event locales and targets therein. The output 6 can be transmitted by cable, optical fiber or wirelessly. It can also be displayed and/or recorded.

The basic processing unit required to operate the system in real-time is shown in Fig. 3. This module comprises:

- a frame grabber for colour image acquisition;

1 a plurality of image memories;  
2 a fast parallel processor;  
3 a program memory;  
4 data banks of synthetic images to be substituted  
5 and of locale's lay-outs and target icons;  
6 a man/machine interface for control and for local  
7 display and recording; and  
8 an image digital to analog converter.

9 The above apparatus is used to automatically  
10 locate in real time in each video frame, suitable areas  
11 within a stadium which have physical displays or might  
12 be suitable for embodying such displays, and to  
13 substitute for such physical displays, or introduce  
14 into such areas, virtual images which are stored in the  
15 memory of the system to serve as advertisements in the  
16 background.

17 These electronic inserted images will be seen  
18 by viewers as if they are physical displays located in  
19 a stadium and all action taking place in front of the  
20 actual physical display will appear to the viewer to be  
21 taking place in front of the virtual image as well.

22 Fig. 4 illustrates an on-line system in  
23 accordance with an aspect of this invention consisting  
24 of a video camera 10, video processing unit 12 and  
25 work station 14 that provides the required man/machine  
26 interface.

27 Fig. 5 illustrates a basic off-line system in  
28 accordance with one aspect of this invention. In this  
29 case, a video tape 20, a video cassette recorder or a  
30 video disk is the input rather than a TV camera and  
31 this is processed by the processing unit 22 and work  
32 station 24 to provide a video tape output 26 with  
33 substituted images.

34 Fig. 6 illustrates yet another application of  
35 the system of this invention, namely a cable TV center.  
36 The center 30 receives transmissions from stations 32  
37 and 34. These transmissions are processed by the  
38 processing unit 22 and work station 24 and broadcast

1 with substituted advertisements to subscribers from the  
2 center 30.

3           Although a preferred system according to this  
4 invention superimposes blocking objects and shadows on  
5 the virtual images, a less sophisticated and much  
6 cheaper system is also intended as part of this  
7 invention, and that is a system where virtual images  
8 are exchanged for physical without relating to blocking  
9 objects.

10           Such a system can be quite useful for  
11 substituting images in unblocked regions, for example  
12 high up in a stadium.

13           Although a preferred embodiment of the  
14 present invention automatically detects and recognizes  
15 a given billboard in each TV frame, a less  
16 sophisticated system is also intended as part of this  
17 invention. In such a less sophisticated system the  
18 selection of a given sign to be substituted is done  
19 "manually" by a pointer such as a light pen or a cursor  
20 (operated by a mouse) with a human operator in the  
21 loop.

22           This system is mainly off-line. When it is  
23 used on-line in real time it will be very difficult for  
24 the operator to perform the pointing task since in a  
25 typical scenario the sign is continuously visible for  
26 only short periods of a few seconds each.

27           In such a mode of operation the replacement  
28 will nevertheless be perceptible to the TV viewers.  
29 This annoys the spectators and in many cases is not  
30 permitted by the TV networks.

31           From the above description of the invention,  
32 it is apparent that the system, method and apparatus  
33 described above can have many applications. Thus, it  
34 is also possible to introduce virtual images, such as  
35 slogans or graphic advertisement, on the uniforms of  
36 players, particularly when a player is shown in close-  
37 up. In such a case, the outline of the player, or at  
38 least his shirt or helmet, would be the target for

1 implanting a virtual image.

2           Another possible application is the automatic  
3 generation of continuous video films showing only  
4 sequences wherein specific targets, which have been  
5 pre-selected, appear to the exclusion of sequences  
6 where these targets do not appear. Such video films  
7 can be useful for analyzing and monitoring the activity  
8 of specific targets, for example individual players and  
9 their performance throughout an entire team game. This  
10 enables tracking each individual throughout an entire  
11 game without having to replay the entire cassette for  
12 each player.

13           Another application of this invention is to  
14 generate statistical data of targets such as  
15 advertisements, for example the number of times and  
16 accumulated period that an advertisement appears on  
17 the screen, and to debit accordingly.

18           The implanted image can be in the form of a  
19 fixed, blinking or scrolling image, or it may be an  
20 animated film or video clip.

21           Fig. 7 is a simplified block diagram of a  
22 real time system for advertisement site detection and  
23 advertisement incorporation, constructed and operative  
24 in accordance with a preferred embodiment of the  
25 present invention.

26           The apparatus of Fig. 7 includes a video  
27 input source 100, such as a video camera, video  
28 cassette, broadcast, video disk, or cable transmission,  
29 which is connected, via a suitable connector, with a  
30 field grabber 110, preferably, or alternatively with a  
31 frame grabber. Henceforth, use of the term "field  
32 grabber" is intended to include frame grabbers.

33           The field grabber 110 provides grabbed and  
34 digitized fields to a parallel processor and controller  
35 120, described in more detail below with reference to  
36 Fig. 8, which is preferably associated with a video  
37 display 130 which provides an interactive indication to  
38 a user of advertisement site detection and adver-



1 tishment incorporation operations of the system.  
2 Preferably a light pen 140 is associated with the video  
3 display 130.

4         According to an alternative embodiment of the  
5 present invention, the system receives an indication  
6 from a user of the presence in the field of view of one  
7 or more advertisements to be replaced and of the  
8 location/s thereof. The user input may, for example, be  
9 provided by means of a light pen 140. The indication  
10 provided by the user may comprise a single indication  
11 of an interior location of the advertisement, such as  
12 the approximate center of the advertisement or may  
13 comprise two or four indications of two opposite  
14 vertices or all four vertices, respectively, of an  
15 advertisement to be replaced.

16         Optionally, the user also provides an  
17 indication of the contents of the advertisement. For  
18 example, a menu of captions identifying advertisements  
19 to be replaced, may be provided on the video display  
20 130 adjacent or overlaying a display of the playing  
21 field and the user can employ the light pen to identify  
22 the appropriate caption.

23         An advertisement images and advertisement  
24 arrangement database 150 is provided which may be  
25 stored in any suitable type of memory such as computer  
26 memory or secondary memory, such as a hard disk. The  
27 advertisement image and arrangement database 150  
28 typically stores a plurality of advertisement images,  
29 typically still images, including images to be replaced  
30 and/or images to be incorporated into the image of the  
31 playing field, either replacing an existing  
32 advertisement or in a location not presently occupied  
33 by an advertisement.

34         The database 150 may also include an  
35 indication of the arrangement of a plurality of  
36 advertisements to be replaced, if the arrangement is  
37 known ahead of time. Typically, the indication of the  
38 arrangement does not include an indication of the

1 location of each advertisement relative to the playing  
2 field, but instead includes an indication of the order  
3 in which the advertisements to be replaced will be  
4 arranged in the field. For example, a sequence of 20  
5 side-by-side advertisements may be arranged around  
6 three sides of a playing field. The database 150 may  
7 then include an indication of the sequence in which the  
8 advertisements are arranged.

9           Advertisement images in the database 150 may  
10 be provided by field grabber 110 or from any suitable  
11 advertisement image source 160, such as but not limited  
12 to an image generating unit such as a image processing  
13 workstation, a scanner or other color reading device,  
14 any type of storage device, such as a hard disk, a CD  
15 ROM driver, or a communication link to any of the  
16 above.

17           The video output of the system may be  
18 provided via a suitable connector to suitable equipment  
19 for providing wireless or cable transmission to  
20 viewers.

21           Fig. 8 is a simplified block diagram of the  
22 parallel processor and controller 120 of Fig. 7. The  
23 parallel processor/controller 120 preferably includes  
24 an advertisement site detection/content identification  
25 unit 170, a plurality of parallel tracking modules 180,  
26 an occlusion analysis and advertisement incorporation  
27 unit 190, a video encoder 200 and a controller 210.

28           The advertisement site detection/content  
29 identification unit 170 of Fig. 8 may be implemented  
30 based on a suitable plurality of suitable image  
31 processing boards, such as Ariel Hydra boards,  
32 commercially available from Ariel, USA. Each of these  
33 preferably incorporates four TMS320C40 digital signal  
34 processors, a DRAM of 64 MB, an SRAM of 1 MB, and a VME  
35 bus interface. A specially designed coprocessor is  
36 preferably added to these boards to perform the  
37 segmentation task. The image processing boards are  
38 programmed based on the advertisement site detection

1 and content identification methods of Figs. 11 and 12  
2 on which Appendix A is based in part. For example, the  
3 appropriate portions of the listing of Appendix A may  
4 be converted into Assembler and the resulting code may  
5 be loaded into the digital signal processor of the  
6 image processing board.

7       Each of parallel tracking modules 180 may be  
8 implemented based on one or more image processing  
9 boards, such as Ariel Hydra boards, commercially  
10 available from Ariel, USA. Each of these preferably  
11 incorporates four TMS320C40 digital signal processors,  
12 a DRAM of 64 MB, an SRAM of 1 MB, and a VME bus  
13 interface. The image processing boards are programmed  
14 for parallel operation based on the tracking method of  
15 Fig. 14 on which Appendix A is based in part. For  
16 example, the appropriate portions of the listing of  
17 Appendix A may be converted into Assembler and the  
18 resulting code may be loaded into the digital signal  
19 processor of the image processing board.

20       The occlusion analysis and advertisement  
21 incorporation unit 190 may also be based on one or more  
22 texture mapping boards such as the Fairchild's Thru-D  
23 boards with the appropriate bus bridges, programmed  
24 based on the occlusion analysis and advertisement  
25 incorporation methods of Figs. 15 and 16 on which  
26 Appendix A is based in part. For example, the  
27 appropriate portions of the listing of Appendix A may  
28 be converted into Assembler and the resulting code may  
29 be loaded into the processor of the texture mapping  
30 board.

31       Video encoder 200 is operative to perform D/A  
32 conversion.

33       Controller 210 may, for example, comprise a  
34 486 PC programmed based on the control method of Figs.  
35 10A - 10B on which Appendix A is based in part. For  
36 example, the appropriate portions of the listing of  
37 Appendix A may be Intel 486 PC processor.

38       Fig. 9 is a simplified block diagram of an

1 alternative embodiment of a real time system for  
2 advertisement site detection and advertisement  
3 incorporation. In the apparatus of Fig. 9, a  
4 conventional workstation 212, having its own video  
5 display 220 and its own field grabber (not shown), such  
6 as a Silicon Graphics Onyx workstation loaded with a  
7 video board and a suitable software, replaces the  
8 following units of Fig. 7: field grabber 110, the  
9 parallel processor and controller 120 other than the  
10 advertisement site detection and content identification  
11 unit 170 and tracking modules 180 thereof, the video  
12 display, and the database 150.

13 The software for the workstation may be based  
14 on the Appendix A implementation of the method of Figs.  
15 10A - 10B, suitably converted into the workstation's  
16 environment, however some of the functions of Appendix  
17 A are preferably omitted. Specifically:

18 a. the advertisement site detection and  
19 tracking functions, corresponding to the segmentation,  
20 advertisement content identification and tracking steps  
21 320, 330 and 310 respectively of Figs. 10A - 10B are  
22 omitted and are instead implemented in real time by  
23 dedicated hardware 230 in Fig. 9; and

24 b. The texture mapping functions (second and  
25 third steps of Fig. 16) which preferably form part of  
26 the advertisement incorporation function, are  
27 preferably omitted and are, instead, performed by the  
28 texture mapping functions provided by the workstation  
29 itself.

30 The dedicated hardware 230 of Fig. 9 may be  
31 similar to the advertisement site detection/content  
32 identification unit 170 and parallel tracking modules  
33 180 of Fig. 8.

34 Appendix A is a computer listing of a non-  
35 real time software implementation of the present  
36 invention which is operative, for example, on a 486 PC  
37 in conjunction with a conventional frame grabber such  
38 as an Imaging MFG board. The method of Appendix A is

1 now described with reference to Figs. 10A - 16.

2           Fig. 10A is a simplified flowchart of a  
3 preferred method of operation of the parallel processor  
4 and controller 120 of Fig. 7, when only a single  
5 advertisement site is to be identified and only a  
6 single advertisement image is to be incorporated at  
7 that site.

8           Fig. 10B is a simplified flowchart of a  
9 preferred method of operation of the parallel processor  
10 and controller 120 of Fig. 7, when a plurality of  
11 advertisement sites is to be identified and a  
12 corresponding plurality of advertisement images, which  
13 may or may not differ in content, is to be incorporated  
14 at those sites respectively.

15           The method of Fig. 10B typically includes the  
16 following steps, which are similar to the steps of Fig.  
17 10A which are therefore not described separately for  
18 brevity:

19           STEP 290: A digitized video field is  
20 received from the field grabber 110 of Fig. 1.

21           STEP 300: A decision is made as to whether or  
22 not at least one advertisement in the current field was  
23 also present in the previous field (and televised by  
24 the same camera). If so, the current field is termed a  
25 "consecutive" field and the segmentation, content  
26 identification and localization steps 320, 330 and 340  
27 preferably are replaced only by a tracking step 310. If  
28 not, the current field is termed a "new" field.

29           If the field is a "consecutive" field, the  
30 plurality of advertisements is tracked (step 310),  
31 based on at least one advertisement which was present  
32 in a previous field, since the present field is a  
33 "consecutive" field.

34           If the field is a "new" field, the  
35 advertisement site at which an advertisement is to be  
36 incorporated is identified in steps 320, 330 and 340. A  
37 loop is performed for each advertisement from among the  
38 plurality of advertisements to be processed.

1 Preferably, the segmentation and content identification  
2 steps 320 and 330 are performed only for the first  
3 advertisement processed.

4 In step 320, a pair of generally parallel  
5 lines is typically detected and the image of the field  
6 is segmented. Specifically, the portion of the field  
7 located within the two detected parallel lines, which  
8 typically correspond to the top and bottom boundaries  
9 of a sequence of advertisements, is segmented from the  
10 remaining portion of the field.

11 Typically, the segmentation step 320 is  
12 operative to segment advertisements regardless of:  
13 their perspective relative to the imaging camera, the  
14 zoom state of the imaging camera lens, the location of  
15 the advertisement in the field of view (video field),  
16 the angular orientation of the imaging camera relative  
17 to the ground and the location of the TV camera.

18 The segmentation step 320 is typically  
19 operative to identify an empty or occupied  
20 advertisement site on a basis other than location, such  
21 as but not limited to any of the following, separately  
22 or in any combination:

23 a. Geometrical attributes of the advertisement's  
24 boundary such as substantially parallel top and bottom  
25 boundaries or such as four vertices arranged in a  
26 substantially rectangular configuration;

27 b. A color or a combination of colors or a color  
28 pattern, which are known in advance to be present in  
29 the advertisement image.

30 c. The spatial frequencies band of the  
31 advertisement image, which is typically known in  
32 advance. Typically, the known spatial frequencies band  
33 is normalized by the height of the advertisement which  
34 may, for example, be derived by computing the distance  
35 between a pair of detected horizontal lines which are  
36 known to be the top and bottom boundaries of the  
37 advertisement sequence.

38 In step 330, the content of the portion

1 between the two substantially parallel lines is matched  
2 to a stored representation of an advertisement to be  
3 replaced.

4 Steps 320 and 330 allow advertisement sites  
5 to be identified and the content thereof to be matched  
6 to a stored model thereof, even if cuts (transitions,  
7 typically abrupt, between the outputs of a plurality of  
8 cameras which are simultaneously imaging the sports  
9 event) occur during the sports event. Typically, at  
10 each cut, steps 320 and 330 are performed so as to  
11 identify the advertisement within the first few fields  
12 of the cut. Until the next cut occurs, the identified  
13 advertisement is typically tracked (step 310).

14 In step 340, the advertisement is localized  
15 at subpixel accuracy.

16 Finally, for each advertisement, occlusion  
17 analysis is performed (step 350) and the replacing  
18 advertisement is incorporated in the advertisement site  
19 (step 360). Alternatively, the occlusion analysis and  
20 advertisement incorporation steps are replaced by an  
21 advertisement enhancement step in which the existing  
22 advertisement is enhanced, using conventional edge  
23 sharpening techniques, rather than being replaced.

24 Optionally, a fee accumulation step 362 is  
25 performed, typically after occlusion analysis step 350.  
26 In the fee accumulation step, a fee for each  
27 advertisement is accumulated. The fee may be computed  
28 on any suitable basis. For example, the fee may be  
29 determined by counting the total amount of time for  
30 which the advertisement was displayed and for which at  
31 least 50% of the advertisement was unoccluded, and  
32 multiplying by a fixed dollar rate per time unit.  
33 Alternatively, the proportion of the unoccluded area of  
34 the advertisement may be computed for each time  
35 interval, such as each second. Optionally, the display  
36 time or the sum over time of the displayed area may be  
37 adjusted to take into account the game's progress. For  
38 example, the display time or the sum over time of the

1 displayed area may be multiplied by an externally  
2 provided index indicating the tension level of the game  
3 during display of the advertisement. High tension level  
4 may, for example, mean that the game has gone into  
5 overtime or that a significant event, such as a goal,  
6 has occurred during display or just before display.  
7 Alternatively, the tension level index may be provided  
8 by the system itself. For example, a voice recognition  
9 unit may recognize significant words uttered by the  
10 sports commentator, such as the word "goal".

11 According to an alternative embodiment of the  
12 present invention, the segmentation and advertisement  
13 content identification steps 320 and 330 respectively  
14 may be omitted if physical landmarks identifying the  
15 locations of advertisements to be replaced whose  
16 contents is known in advance, are positioned and  
17 captured ahead of time in the playing field.

18 Fig. 11 is a simplified flowchart of a  
19 preferred method for performing the segmentation step  
20 320 of Figs. 10A and 10B.

21 The method of Fig. 11 preferably includes the  
22 following steps:

23 STEP 380: A new field is received and the  
24 resolution thereof is preferably reduced since the  
25 forgoing steps may be performed adequately at a lower  
26 resolution. for example, a low pass filter may be  
27 employed to reduce a 750 x 500 pixel field to 128 x 128  
28 pixels.

29 STEP 390: Optionally, the low resolution  
30 image is smoothed, e.g. by median filtering or low pass  
31 filtering, so as to remove information irrelevant to  
32 the task of searching for long or substantially  
33 horizontal lines.

34 STEP 400: Edges and lines (two-sided edges)  
35 are detected, using any suitable edge detection method  
36 such as the Canny method, described by J.F. Canny in "A  
37 computational approach to edge detection", IEEE Trans.  
38 Pattern Analysis and Machine Intelligence, Vol. 8, pp.



1 679-698, November, 1986.

2 STEP 404: The edges detected in step 400 are  
3 thinned and components thereof are connected using  
4 conventional techniques of connectivity analysis. The  
5 edges are thresholded so as to discard edges having too  
6 small a gradient.

7 STEP 408: The edges detected in steps 400 and  
8 410 are compared pairwise so as to find strips, i.e.  
9 pairs of parallel or almost parallel lines which are  
10 relatively long. If there are no such pairs, the method  
11 terminates.

12 STEP 412: Find the spatial frequency spectrum  
13 within each strip and reject strips whose spatial  
14 frequency contents are incompatible with the spatial  
15 frequency band expected for advertisements. Typically,  
16 the rejection criterion is such that more than one  
17 strip, such as 3 or 4 strips, remain.

18 STEP 416: Rank the remaining strips and  
19 select the highest ranking strip. The rank assigned to  
20 a strip depends on the probability that the strip  
21 includes advertisements. For example, the strip in the  
22 lowest location in the upper half of the field is given  
23 higher rank than strips above it, because the strips  
24 above it are more likely to be images of portions of  
25 the stadium. The lowest located strip is more likely to  
26 be the advertisements which are typically positioned  
27 below the stadium.

28 Strips adjacent the bottom of the field are  
29 given low rank because the advertisements would only be  
30 imaged toward the bottom of the video field if the  
31 playing field is not being shown at all, which is  
32 unlikely.

33 Fig. 12 is a simplified flowchart of a  
34 preferred model matching method for performing the  
35 advertisement content identification step 330 of Figs.  
36 10A and 10B. Alternatively, advertisement content  
37 identification may be provided by a user, as described  
38 above with reference to Fig. 1.

1           The method of Fig. 12 is preferably performed  
2 in low resolution, as described above with reference to  
3 step 380 of Fig. 11. The method of Fig. 12 preferably  
4 includes the following steps:

5           STEP 420: The forgoing steps 424, 430, 436,  
6 440, 444 and 452 are performed for each almost  
7 parallel strip identified in segmentation step 320 of  
8 Fig. 11.

9           STEP 424: The distance and angle between the  
10 two lines of each strip is computed and the scale and  
11 approximate perspective at which the strip was imaged  
12 is determined therefrom.

13          STEP 430: During set-up, each advertisement  
14 model is divided into a plurality of windows. Steps  
15 436, 440 and 444 are performed for each window of each  
16 advertisement model. For example, if there are 5 models  
17 each partitioned into 6 windows, this step is performed  
18 30 times.

19          STEP 436: A one-dimensional similarity search  
20 is carried out for the suitably scaled current model  
21 window  $k$ , along the current almost parallel strip.  
22 Typically, a cross-correlation function may be computed  
23 for each pixel along the current strip.

24          STEP 440: The cross-correlation function  
25 values obtained in step 436 are thresholded. For  
26 example, values exceeding 0.6 may be assigned the value  
27 1 (correlation) whereas values under 0.6 may be  
28 assigned the value 0 (no correlation). The 1's are  
29 weighted, depending on the "significance" of their  
30 corresponding windows. The "significance" of each  
31 window is preferably determined during set-up such that  
32 windows containing more information are more  
33 "significant" than windows containing little  
34 information.

35          STEP 444: At this stage, weighted thresholded  
36 cross-correlation function values have been computed  
37 which represent the results of matching the contents of  
38 each position along the strip (e.g. of each of a  
28

1 plurality of windows along the strip which are spaced  
2 at a distance of a single pixel) to each window of each  
3 model advertisement known to occur within the strip.

4         The weighted thresholded cross-correlation  
5 function values are accumulated per all windows  
6 composing a model sign or a model strip.

7         STEP 452: A decision is made as to the  
8 approximate location of the sequence of advertising  
9 models, within the strip. It is appreciated that, once  
10 the location of one advertisement model has been  
11 determined, the locations of the other advertisement  
12 models in the same sequence are also determined,  
13 knowing the scale and approximate perspective of the  
14 imaged strip.

15         Fig. 13 is a simplified flowchart of a  
16 preferred method for performing the precise  
17 localization step 340 of Figs. 10A and 10B. In Fig. 13,  
18 the advertisement model which was approximately  
19 localized by the method of Fig. 12, is localized with  
20 subpixel accuracy. Accurate localization is typically  
21 performed only for new fields. For "consecutive"  
22 fields, the advertisement's location is preferably  
23 measured by video tracking.

24         The method of Fig. 13 preferably includes the  
25 following steps:

26         STEP 460: From Fig. 12, the following  
27 information is available per advertisement detected:  
28 one location within the advertisement, such as one  
29 vertex thereof, the advertisement scale height in the  
30 image and its approximate perspective. This information  
31 is employed to compute the four vertices of each  
32 detected advertisement.

33         STEP 464: A perspective transformation is  
34 computed which describes how to "transform" the  
35 typically rectangular model into the detected  
36 advertisement area which is typically non-rectangular  
37 due to its pose relative to the imaging camera.

38         STEP 468: The contents of each of a plurality

1 of model tracking windows to which the model is divided  
2 during set up, is mapped into the video field, using  
3 the perspective transformation computed in step 464.

4 STEP 470: Steps 472 and 476 are performed for  
5 each of the model tracking windows.

6 STEP 472: The current model tracking window  
7 is translated through a search area defined in the  
8 video field. For each position of the model tracking  
9 window within the search area, a similarity error  
10 function (like cross-correlation or absolute sum of  
11 differences) is computed. Typically, the model tracking  
12 window has 8 x 8 or 16 x 16 different positions within  
13 the search area.

14 STEP 476: The minimum similarity error  
15 function for the current model tracking window is  
16 found. Preferably, the minimum is found at subpixel  
17 accuracy, e.g. by fitting a two-dimensional parabola to  
18 the similarity error function generated in step 472 and  
19 computing the minimum of the parabola. This minimum  
20 corresponds to the best position, at "subpixel  
21 accuracy", for the current model tracking window within  
22 the video field.

23 If (STEP 480) the similarity error function  
24 minima are high for all tracking windows, i.e. none of  
25 the tracking windows can be well matched to the video  
26 field, then (STEP 482) processing of the current frame  
27 is terminated and the method of Fig. 10A, from step 320  
28 onward, is performed on the following frame.

29 STEP 484: Tracking windows which have a high  
30 similarity error function minimum are rejected.  
31 Typically, approximately 30 tracking windows remain.

32 STEP 488 is a stopping criterion determining  
33 whether or not to perform another iteration of  
34 localization by matching tracking windows. Typically,  
35 if the tracking windows' centers are found to converge,  
36 relative to the centers identified in the last  
37 iteration, the process is terminated. Otherwise, the  
38 method returns to step 464.

1           STEP 490: Once the tracking window locations  
2 have converged, the perspective transformation between  
3 the images advertisement and its model is recomputed.

4           Fig. 14 is a simplified flowchart of a  
5 preferred method for performing the tracking step 310  
6 of Figs. 10A and 10B. The method of Fig. 14 preferably  
7 includes the following steps:

8           STEP 492: A perspective transformation is  
9 performed on the model tracking windows and the  
10 contents thereof are mapped into the video field. This  
11 step employs the system's knowledge of the location of  
12 the advertisement in the previous field and,  
13 preferably, predicted scanning speed of the camera  
14 imaging the sports event.

15           STEP 496: Steps 498 and 500, which may be  
16 similar to steps 472 and 476, respectively, of Fig. 13,  
17 are performed for each model tracking window.

18           STEPS 508 AND 512 may be similar to steps 488  
19 and 490 of Fig. 13.

20           STEP 510: If the window center locations do  
21 not yet converge, step 492 is redone, however, this  
22 time, the texture mapping is based upon the perspective  
23 transformation of the previous iteration.

24           STEP 520: The coefficients of the perspective  
25 transformation are preferably temporally smoothed,  
26 since, due to the smoothness of the camera's scanning  
27 action, it can be assumed that discontinuities are  
28 noise.

29           Fig. 15 is a simplified flowchart of a  
30 preferred method for performing the occlusion analysis  
31 step 350 of Figs. 10A and 10B. The method of Fig. 15  
32 preferably includes the following steps:

33           STEP 530: The advertisement image in the video  
34 field is subtracted from its perspective transformed  
35 model, as computed in step 512 of Fig. 14 or, for a new  
36 field, in step 390 of Fig. 13.

37           STEP 534: Preferably, the identity of the  
38 advertisement image and the stored advertisement is

1 verified by inspecting the difference values computed  
2 in step 530. If the advertisement image and the stored  
3 advertisement are not identical, the current field is  
4 not processed any further. Instead, the next field is  
5 processed, starting from step 320 of Fig. 10B.

6 STEP 538: The internal edge effects are  
7 filtered out of the difference image computed in step  
8 530 since internal edges are assumed to be artifacts.

9 STEP 542: Large non-black areas in the  
10 difference image are defined to be areas of occlusion.

11 STEP 546: The occlusion map is preferably  
12 temporally smoothed since the process of occlusion may  
13 be assumed to be continuous.

14 Fig. 16 is a simplified flowchart of a  
15 preferred method for performing the advertisement  
16 incorporation step 360 of Figs. 10A and 10B. The method  
17 of Fig. 16 preferably includes the following steps:

18 STEP 560: The resolution of the replacing  
19 advertisement model, i.e. the advertisement in memory,  
20 is adjusted to correspond to the resolution in which  
21 the advertisement to be replaced was imaged. Typically,  
22 a single advertisement model is stored in several  
23 different resolutions.

24 STEP 570: The replacing advertisement is  
25 transformed and texture mapped into the video field  
26 pose, using tri-linear interpolation methods. This step  
27 typically is based on the results of step 512 of Fig.  
28 14 or, for a new field, on the results of step 390 of  
29 Fig. 13.

30 STEP 580: Aliasing effects are eliminated.

31 STEP 590: The replacing pixels are keyed in  
32 according to an occlusion map. The values of the  
33 replacing pixels may either completely replace the  
34 existing values, or may be combined with the existing  
35 values, as by a weighted average. For example, the  
36 second alternative may be used for edge pixels whereas  
37 the first alternative may be used for middle pixels.

38 Fig. 17 is a simplified block diagram of

1 camera monitoring apparatus useful in conjunction with  
2 a conventional TV camera and with the advertisement  
3 site detection/incorporation apparatus of Fig. 7. If  
4 the parallel processor and controller of Fig. 7 is as  
5 illustrated in Fig. 8, the apparatus of Fig. 17 is not  
6 required and instead, a conventional TV camera may be  
7 employed. However, in the alternative, the automatic  
8 detection and content identification features of the  
9 system may be eliminated, by eliminating unit 170 of  
10 Fig. 8. In this case, the apparatus of Fig. 17 is  
11 preferably provided in operative association with the  
12 TV camera at the stadium or playing field.

13           The apparatus of Fig. 17 provides camera  
14 information, including the identity of the "on-air"  
15 camera, its lens zoom state and the direction of its  
16 FOV center. This information may be employed, in  
17 conjunction with known information as to the positions  
18 and contents of advertisements in the stadium, in order  
19 to detect, identify and even roughly track each  
20 advertisement.

21           The apparatus of Fig. 17 includes:

- 22 (a) a plurality of conventional TV cameras 600 of  
23 which one is shown in Fig. 17;
- 24 (b) for each camera 600, a camera FOV (field of  
25 view) center direction measurement unit 610 at least a  
26 portion of which is typically mounted on the TV camera  
27 600 pedestal;
- 28 (c) for each camera 600, a camera lens zoom state  
29 monitoring unit 620 which is typically mounted on the  
30 TV camera 600 pedestal. The monitoring unit 620  
31 receive an output indication of the zoom state  
32 directly from the zoom mechanism of the camera;
- 33 (d) an "on-air" camera identification unit 630  
34 operative to identify the camera, from among the  
35 plurality of TV cameras 600, which is being broadcast.  
36 This information is typically available from the  
37 broadcasting system control unit which typically re-  
38 ceives manual input selecting an on-air camera, from a

1 producer; and  
2 (e) a camera information video mixer 640  
3 operative to mix the output of units 610, 620 and 630  
4 onto the broadcast. Any suitable mixing may be  
5 employed, such as mixing onto the audio channel, mixing  
6 onto the time code, or mixing onto the video signal  
7 itself.

8 The camera FOV direction measurement unit 610  
9 may be implemented using any of the following methods,  
10 inter alia:

11 a. On-camera NFM (North Finding Module) in  
12 conjunction with two inclinometers for measuring the  
13 two components of the local gravity vector angle with  
14 respect to the FOV center direction;

15 b. GPS- (Global Position System) based direction  
16 measurement system;

17 c. Triangulation -- positioning two RF sources  
18 at two known locations in the playing field or stadium  
19 and an RF receiver on the camera;

20 d. an on-camera boresighted laser designator in  
21 combination with an off-camera position sensing  
22 detector operative to measure the direction of the beam  
23 spot generated by the laser designator.

24 Fig. 18 is a simplified flowchart of an  
25 optional method for processing the output of the  
26 occlusion analysis process of Fig. 15 in order to take  
27 into account images from at least one off-air camera.  
28 If the method of Fig. 18 is employed, a video  
29 compressor and mixer 700 are provided in operative  
30 association with the TV cameras which are imaging the  
31 event at the playing field or stadium, as shown in Fig.  
32 2. The output of the compressor and mixer 700,  
33 comprising compressed images of the playing field as  
34 imaged by all of the TV cameras other than the TV  
35 camera which is "on-air", blended with the broadcast  
36 signal, is broadcast to remote advertisement site  
37 detection/incorporation systems such as that  
38 illustrated in Fig. 7. The transmission provided by



1 compressor and mixer 700 of Fig. 2 is first decoded and  
2 decompressed in step 710 of Fig. 18.

3 STEP 720: Steps 730, 740 and 750 are repeated  
4 for each advertisement site imaged by the "on air"  
5 camera.

6 STEP 730: Although it is possible to employ  
7 information from more than one of the "off-air"  
8 cameras, preferably, only a single "off air" camera is  
9 employed to process each advertisement site and the  
10 single "off-air" camera is selected in step 730. For  
11 example, if the apparatus of Fig. 17 is provided, the  
12 output of camera FOV direction measurement unit 610 for  
13 each "off-air" camera may be compared in order to  
14 identify the "off-air" camera whose FOV direction is  
15 maximally different from the FOV direction of the "on-  
16 air" camera. Alternatively, particularly if the appa-  
17 ratus of Fig. 17 is omitted, a single "off-air" camera  
18 may be selected by performing preliminary analysis on  
19 the images generated by each of the "off-air" cameras  
20 in order to select the most helpful "off-air" camera.  
21 For example, the images generated by each "off-air"  
22 camera may be matched to the stored representation of  
23 the advertisement currently being processed. Then, the  
24 actual image may be warped and then subtracted from the  
25 stored representation for each "off-air" camera in  
26 order to obtain an estimate of the occlusion area for  
27 that camera and that advertisement. The camera with the  
28 minimal occlusion area may then be selected.

29 STEP 740: The advertisement image of the  
30 selected "off-air" camera is warped onto the  
31 advertisement site as imaged by the "on-air" camera.

32 STEP 750: The warped "off-air" advertisement  
33 image is subtracted from the "on-air" image and the  
34 difference image is filtered in order to compute the  
35 boundary of the occluding object at pixel-level  
36 accuracy.

37 According to a preferred embodiment of the  
38 present invention, the advertisement to be incorporated

1 in a particular location in the playing field or other  
2 locale may vary over time. This variation may be in  
3 accordance with a predetermined schedule, or in  
4 accordance with an external input. For example, a  
5 speech recognition unit may be provided which is  
6 operative to recognize key words, such as the word  
7 "goal" or the word "overtime", on the audio channel  
8 accompanying the video input to the system. In this  
9 way, an advertisement may be scheduled to be  
10 incorporated at particular times, such as just after a  
11 goal or during overtime.

12 In the present specification, the term  
13 "advertisement site" refers to a location into which an  
14 advertisement is to be incorporated. If an existing  
15 advertisement occupies the advertisement site, the new  
16 advertisement replaces the existing advertisement.  
17 However, the advertisement site need not be occupied by  
18 an existing advertisement. The term "occluded"  
19 refers to an advertisement site which is partially or  
20 completely concealed by an object, typically a moving  
21 object, in front of it.

22 A particular feature of the present invention  
23 is that, when it is desired to track an advertisement  
24 site within a larger image, the entire image is not  
25 tracked, but rather only the advertisement site itself.

26 Another particular feature is that "special"  
27 advertisements may be provided, such as moving,  
28 blinking or otherwise varying advertisements, video  
29 film advertisements, advertisements with changing  
30 backgrounds, and advertisements with digital effects.

31 It is appreciated that the particular  
32 embodiment described in Appendix A is intended only to  
33 provide an extremely detailed disclosure of the present  
34 invention and is not intended to be limiting.

35 The applicability of the apparatus and  
36 methods described above is not limited to the  
37 detection, tracking and replacement or enhancement of  
38 advertisements. The disclosed apparatus and methods

1 may, for example, be used to detect and track moving  
2 objects of central interest, as shown in Fig. 19, such  
3 as focal athletes and such as balls, rackets, clubs and  
4 other sports equipment. The images of these moving  
5 objects may then be modified by adding a "trail"  
6 including an advertisement such as the logo of a  
7 manufacturer.

8           It is appreciated that various features of  
9 the invention which are, for clarity, described in the  
10 contexts of separate embodiments may also be provided  
11 in combination in a single embodiment. Conversely,  
12 various features of the invention which are, for  
13 brevity, described in the context of a single  
14 embodiment may also be provided separately or in any  
15 suitable subcombination.

16           It will be appreciated by those skilled in  
17 the art that the invention is not limited to what has  
18 been shown and described hereinabove. Rather, the scope  
19 of the invention is defined solely by the claims which  
20 follow:

21  
22  
23  
24  
25  
26  
27  
28  
29  
30  
31  
32  
33  
34  
35  
36  
37  
38

APPENDIX A

```

#include "comp_fnc.h"
int __cdecl compare_short_pnt ( const void *elem1,const void *elem2 )
{int i; double t1,t2;
 double a;
 t1=**(double**)elem1;t2=**(double**)elem2;
 a=t1-t2;
 i=(a?((a<0)?1:-1):0);
 return i;
}
//=====

```

```

int __cdecl compare_GOOD_DIR_LINE ( const void *elem1,const void
*elem2 )
{int i; double t1,t2;
 double a;
 t1=((GOOD_DIR_LINE*)elem1)->Qual;t2=((GOOD_DIR_LINE*)elem2)-
>Qual;
 a=t1-t2;
 i=(a?((a<0)?1:-1):0);
 return i;
}
//=====
=====
//=====
=====

```

```

int __cdecl compare_array_elem ( const void *elem1,const void *elem2 )
{int i;
 double a;
 a=**(double **)elem1-**(double **)elem2;
 i=(a?((a<0)?1:-1):0);
 return i;
}
//=====
=====

```

```

#ifndef COR_FNC
#define COR_FNC
#include "projctn8.h"
#include "pic_mch8.h"
#include "res_mch8.h"
typedef struct
    {double md[3],var[3];} AVERAGE_VEC;

AVERAGE_VEC  average(PCT tag,double Thr,
    COLOR_VEC (*p_funct)(COLOR_RGB p1,double Thresh_mdl));
COLOR_VEC template_conv_1 (PCT target,PCT win,double
    Thr,COLOR_VEC int_v,
        COLOR_VEC (*p_funct)(COLOR_RGB p1,double
    Thresh_mdl));
COLOR_VEC template_abs_diff_1 (PCT target,PCT win,double
    Thr,COLOR_VEC int_v,
        COLOR_VEC (*p_funct)(COLOR_RGB p1,double Thresh_mdl),
        AVERAGE_VEC t,AVERAGE_VEC w);
COLOR_VEC Correlation(COLOR_VEC conv,AVERAGE_VEC
    t,AVERAGE_VEC w,
        COLOR_VEC int_v);
COLOR_VEC Correlation_single_1(COLOR_VEC conv,AVERAGE_VEC
    t,AVERAGE_VEC w,
        COLOR_VEC int_v);
AVERAGE_VEC LineMoments(PCT &T, DIR_LINE Line, short map);
double Quality(AVERAGE_VEC * UpLineQuality);

#endif

```

```
#ifndef LIBR
#define LIBR
#include "projtn8.h"
#include "pic_mch8.h"
#include "res_mch8.h"
//#include "lin_tm7.h"
```

```
#define NAME_LENGTH 40
#define GRAPHMODE _VRES256COLOR
```

```
#endif
```

```
typedef struct {  
    int cols, rows;  
    int bpp;  
} PHDR;
```



```

#ifndef PIC_MCH
#define PIC_MCH
#include <vmemory.h>
#include <graph.h>
#include "projctn8.h"
#define MEDIAN_AREA 49

```

```

typedef unsigned char byte;

```

```

typedef struct
{short r,g,b;}
COLOR_RGB;

```

```

class PCT
{public:
short s_rows,s_cols;
unsigned char __far * buffer_now;
unsigned char __far **pict;

```

```

//=====
=====

```

```

PCT()
{buffer_now=NULL;
pict=NULL;
s_rows=s_cols=0;
}

```

```

void free_PCT()
{int i;
if(!pict) return;
for(i=0;i<s_rows;i++)
free((void __far *)pict[i]);
buffer_now=NULL;
pict=NULL;
}

```

```

void put_str(short y,unsigned char *B)
{
if(y<s_rows)
{buffer_now=pict[y];
int i;
for(i=0;i<3*s_cols;i++)
*buffer_now++=*B++;
}
}

```

```

void put_pix(short y,short x,COLOR_RGB p)
{
if((y<s_rows) && (x<s_cols))
buffer_now = pict[y]+3*x;
PutNextPix(p);
}

```

```

//=====
inline COLOR_RGB get_next(void)
{ COLOR_RGB t;
  t.r=*buffer_now++;
  t.g=*buffer_now++;
  t.b=*buffer_now++;

  return t;
}
//=====
inline COLOR_RGB get_pixel(short y,short x)
{
  if((y<s_rows) && (x<s_cols))
    buffer_now= pict[y]+3*x;
  return get_next();
}
//=====
//=====
void PutNextPix(COLOR_RGB p);
PCT (short n_cols, short n_rows);
void load_template(PCT source,SCR_PNT left_up_scr);
COLOR_RGB get_median_pixel(short y,short x,short neighbour);
int load_file_rgb(const char *name);

};

void sign_present_RGB(PCT pict_scr,SCR_PNT left_up);
PCT sign_storage_rgb(const char *name,struct _videoconfig vc);
COLOR_RGB *make_palette();

short color_num(short r,short g,short b);
void write_sign_rgb(char *name,PCT pict_now);
#endif

```

```

#ifndef PIC_PRO
#define PIC_PRO
#include <stdlib.h>
#include <direct.h>
#include <afx.h>
#include <pic_mch7.h>
#include "filemnp.h"
#define STR_MAX 4
//=====
const SCR_PNT z_0(0,0);
class PRT:public PCT
{public:
//information
    CString PathName;
    CString FRAME_Number;
    CString STRING_name;
    CString SIGN_name;
    short Pos; // Position in the string
    long NumberOfChk,MaxNum;
    double *Charact;
//models
    PRT::~PRT()
    {this->free_PCT();
    Pos=0;
    if(MaxNum)
        delete Charact;
    Charact=NULL;
    MaxNum=NumberOfChk=0;
    }
//-----
    PRT::PRT()
    {NumberOfChk=MaxNum=s_cols=s_rows=0;
    Charact=NULL;pict=NULL;
    }
//-----
    PRT::PRT (short n_cols, short n_rows)
    {*(PCT *)this=PCT::PCT(n_cols,n_rows);
    NumberOfChk=MaxNum=0;
    Charact=NULL;
    }
//=====
int read_proto_SGN(char ext[]=".sgn")
{
    CString new_name(" ",80);

    PathName=MakeName(PathName);
    new_name=PathName+ext;
    char now[80];

```

```

FILE *datfp;
if(! (datfp=fopen((const char*)new_name,"r"))) return 1;
if(fscanf(datfp,"%[^\\n]s")==EOF)goto ERR;
if(fscanf(datfp,"%s",now)==EOF)goto ERR;
if(fscanf(datfp,"%s",now)==EOF)goto ERR;
if(fscanf(datfp,"%s",now)==EOF)goto
ERR;FRAME_Number=now;
if(fscanf(datfp,"%s",now)==EOF)goto
ERR;STRING_name=now;
if(fscanf(datfp,"%s",now)==EOF)goto ERR; SIGN_name=now;
FRAME_Number.MakeUpper();
STRING_name.MakeUpper();
SIGN_name.MakeUpper();
fclose(datfp);
return 0;
ERR:fclose (datfp); return 1;
}
//=====
int proto_storage_rgb(char *name,struct _videoconfig vc)
{*(PCT *)this=sign_storage_rgb(name,vc);
if (!s_cols) return 1;
PathName=MakeName(name);
if (read_proto_SGN())
{free_PCT();
return 1;
}
return 0;
}
//-----
int read_proto_DBC(FILE *datfp)
{
char now[80];
if(fscanf(datfp,"%s",now)==EOF)goto
ERR;PathName=MakeName(now);
if(fscanf(datfp,"%s",now)==EOF)goto
ERR;FRAME_Number=now;
if(fscanf(datfp,"%s",now)==EOF)goto
ERR;STRING_name=now;
if(fscanf(datfp,"%s",now)==EOF)goto ERR; SIGN_name=now;
if(fscanf(datfp,"%d",&(this->s_cols))==EOF)goto ERR;
FRAME_Number.MakeUpper();
STRING_name.MakeUpper();
SIGN_name.MakeUpper();
return 1;
ERR: return 0;
}
//=====
===
int alloc_Charact_dbl(long Num)

```

```

{
    if(!(Charact=new double[Num])) return 1;
    MaxNum=Num; NumberOfChk=0;
    return 0;
}
//-----
void free_Charact()
{delete Charact;
  Charact=NULL;
}
//-----
int read_Charact_dbl(FILE *inp,long Num)
{short i;
  double d;
  if(MaxNum<(NumberOfChk+Num)) return 1;
  for (i=0;i<Num;i++)
  {if(fscanf(inp,"%lf",&d) ==EOF) return 1;
    if(fabs(d)<1.0e-4) d=0;
    Charact[NumberOfChk]=d;
    NumberOfChk++;
  }
  return 0;
}
//-----
double CorrValue(short WNum,short Pnum)
{return (*(Charact+(long)WNum*s_cols+Pnum));
}
//=====
=====
//=====RETURN NUMBER OF STRIPS
int read_target_SGN(SCR_PNT vrt[][4],char ext[]=".sgs")
{int n=0,i,FLG,s;
  CString new_name(' ',80);

  PathName=MakeName(PathName);
  new_name=PathName+ext;
  char now[80];
  FILE *datfp;
  if(!(datfp=fopen((const char*)new_name,"r"))) return 1;
  if(fscanf(datfp,"%*[\^\\n]s")==EOF)goto OUT;
  if(fscanf(datfp,"%s ",now)==EOF)goto OUT;
  if(fscanf(datfp,"%s ",now)==EOF)goto OUT;
  if(fscanf(datfp,"%s ",now)==EOF)goto
  OUT;STRING_name=now;
  if(fscanf(datfp,"%s ",now)==EOF)goto OUT; SIGN_name=now;
  if((s=PathName.ReverseFind('\\'))<0)
    s=PathName.ReverseFind('.');
  FRAME_Number=
    PathName.Right(PathName.GetLength()-s);
}

```

```

        STRING_name.MakeUpper();
        SIGN_name.MakeUpper();
    do{
        for(j=0;j<4;j++)
            if((FLG=fscanf(datfp,"%d %d",&(vrt[n][j].c),&(vrt[n][j].r)))==EOF)
                goto OUT;
        n++;
    }
    while(n<STR_MAX-1);
    OUT:fclose (datfp); return n;
}
//=====
};
//=====

#define UnKnown -1
//=====
=
typedef struct
{
    short n; // voiting numbers
    short pos; // position in string
    double value; //value
} RSLT_old;

//=====
void HistCollect(short NOfWin,short St,short Fin,PRT &Db);
RSLT_old LineEstimation (short TagSize, PRT &Db,short NOfWin,
                        short WSize,double Thr);
int LineInf(const PRT &P, PRT T, short rw, short Xpos,struct _videoconfig vc);
double LinInter( PRT &P,short WNum,short WSize ,double Pt);
void HistThresh(short *H,short *BotThr,short *TopThr,short num);

#endif

```

```

#ifndef PROJCTN
#define PROJCTN
#include <math.h>
#include <graph.h>
typedef struct {
    double x, y, z;
} PNT;

class SCR_PNT
{public:
    short c, r;
    SCR_PNT(){c=0;r=0;}
    SCR_PNT(short x,short y){c=x;r=y;}
};

//PT -> SCR_PNT conversion //
#define PT_SCR( p,scr)  (scr).c=(short)(p).u;(scr).r=(short)(p).v;
// SCR_PNT -> PT converson
#define INT_PT(scr,dbl)  (dbl).u=(double)((scr).c);\
                        (dbl).v=(double)((scr).r);

class PT
{public:
    double u, v;
    PT(double x, double y){u=x; v=y;}
    PT(SCR_PNT p1 ){u=(double)p1.c;v=(double)p1.r;}
    PT (){u=0;v=0;}
};

class LINE_PROJECTION;
class DIR_LINE
{ friend class LINE_PROJECTION;
private:
    double    a,b,c;    // a*u+b*v+c=0
    PT st_pnt, end_pnt;
public:
    DIR_LINE (PT p1, PT p2)
        {st_pnt=p1;
         end_pnt=p2;
    }
#ifdef DEBUG
        _moveto( (short) st_pnt.u,(short) st_pnt.v );
        _lineto( (short) end_pnt.u,(short) end_pnt.v );
#endif
        a=p2.v-p1.v;
        b=p1.u-p2.u;
        c=p1.v*p2.u-p1.u*p2.v;    // a*x+b*y+c=0
    }

// -----
DIR_LINE (SCR_PNT p1, SCR_PNT p2)

```

```

        {st_pnt=PT::PT(p1);
          end_pnt=PT::PT(p2);
#ifdef DEBUG
          _moveto( (short) st_pnt.u,(short) st_pnt.v );
          _lineto( (short) end_pnt.u,(short) end_pnt.v );
#endif
          a=end_pnt.v-st_pnt.v;
          b=st_pnt.u-end_pnt.u;
          c=st_pnt.v*end_pnt.u-st_pnt.u*end_pnt.v; //
          a*x+b*y+c=0
        }

DIR_LINE ()
{st_pnt.u=st_pnt.v=
end_pnt.u=end_pnt.v=a=b=c=0;
}
PT PT_for_P( double p)
{ PT t;
  t.u=st_pnt.u+p*(end_pnt.u-st_pnt.u);
  t.v=st_pnt.v+p*(end_pnt.v-st_pnt.v);
  return t;
}
double U_for_V(double v) {return(a*(-c-b*v)/a:0);}
double V_for_U(double u) {return(b*(-c-a*u)/b:0);}
double a_for_line(){return a;}
double b_for_line(){return b;}
double c_for_line(){return c;}

double RATIO_for_PT(PT p) //relative point position at line
{ double dx,dy,dro;
  dx=end_pnt.u-st_pnt.u;
  dy=end_pnt.v-st_pnt.v;
  if(fabs(dx)>fabs(dy))
    {if(!dx) return 0;
     dro= (p.u-st_pnt.u)/dx;
    }
  else
    {
      if(!dy) return 0;
      dro= (p.v-st_pnt.v)/dy;
    }
  return dro;
}

int Left_Right(int col,int row)
// +1 if point to the left hand from start to end
// 0 otherwise
{
  return(((a*col+b*row+c)>=0)?0:1);
}

```



```

        PT Start_p(){return st_pnt;}
        PT End_p(){return end_pnt;}
friend    int U_dist(DIR_LINE l);
friend PT cross_line(DIR_LINE Line1, DIR_LINE Line2);
friend int INside(DIR_LINE l1,DIR_LINE l2,DIR_LINE l3,DIR_LINE l4,int
col,int row);
};

```

```

class LINE_PROJECTION
{

```

```

    private:

```

```

        double proj_prm; //

```

```

    public:

```

```

        DIR_LINE Rl, Prj;

```

```

        LINE_PROJECTION(){

```

```

            DIR_LINE l1; Rl=l1; Prj=l1;

```

```

            proj_prm=0;

```

```

        }

```

```

        LINE_PROJECTION (DIR_LINE l1, PT p, DIR_LINE pr, PT p_pr);

```

```

        double Ro_for_P(double P)

```

```

        {return ( (P*(1+proj_prm))/(1+P*proj_prm));

```

```

        }

```

```

        double P_for_Ro(double Ro)

```

```

        {return ( Ro/(1+(1-Ro)*proj_prm));

```

```

        }

```

```

friend DIR_LINE Line_for_PT_pr(LINE_PROJECTION line1, PT pr1,

```

```

        LINE_PROJECTION line2, PT pr2);

```

```

friend DIR_LINE Line_for_PT_rl(LINE_PROJECTION line1, PT p1,

```

```

        LINE_PROJECTION line2, PT p2);

```

```

};

```

```

//=====

```

```

class GOOD_DIR_LINE:public DIR_LINE

```

```

{public:

```

```

    double Qual;

```

```

    GOOD_DIR_LINE(){Qual=0;}

```

```

    GOOD_DIR_LINE(SCR_PNT St,SCR_PNT End)

```

```

    {Qual=0;

```

```

    *(DIR_LINE *)this=DIR_LINE::DIR_LINE(St,End);

```

```

    }

```

```

//        void OnDraw(short incr)

```

```

//        { PT st=Start_p(), stp=End_p();

```

```

//        _moveto( (short) st.u,(short) (st.v+incr) );

```

```

//        _lineto( (short) stp.u,(short) (stp.v+incr) );

```

```

//        }

```

```

};

```

```

typedef struct{ LINE_PROJECTION L_left,L_mid,L_right;

```

```
        DIR_LINE I01, I12, I23, I30;  
        DIR_LINE I01_pr, I12_pr, I23_pr, I30_pr;  
    }  
    RECT_RESOLVED;  
#endif
```

```

#ifndef RES_MCH
#define RES_MCH
#include <graph.h>
#include "projctn8.h"
#include "pic_mch8.h"
#define NTSC 0
#define HSI 1
#define New_plan 2
#define RGB 3
#define LUMIN_THR 4
#define IHS 5
//=====

typedef struct
    {double c[3];
      } COLOR_VEC;

void GRAPH_OUT(int ex=0);
int  GRAPHICS_START(struct _videoconfig *vc,short GR_mode);
void match_vertex(SCR_PNT *v);
int  __cdecl c_comp( const void *elem1, const void *elem2);
short interpol(short * s,double x,double y);

COLOR_RGB INTER_pix_color_rgb(PCT p1, PT PT_now);
COLOR_RGB INTER_pix_color_rgb_median(PCT p1, PT PT_now);

const COLOR_VEC NORM_RGB={256,1,1};
const COLOR_VEC NORM_simple={256,256,256};

COLOR_VEC (*PointColorFunct(short M))(COLOR_RGB p1, double
Thresh_md1);

COLOR_VEC color_space_NTSC(COLOR_RGB p1,double Thresh_md1);
COLOR_VEC color_space_RGB(COLOR_RGB p1,double Thresh_md1);
COLOR_VEC color_space_NEW(COLOR_RGB p1,double Thresh_md1);
COLOR_VEC color_space_RGB_simple(COLOR_RGB p1, double
Thresh_md1);
COLOR_VEC color_space_LUMIN_THR(COLOR_RGB p1, double
Thresh_md1);
COLOR_VEC color_space_IHS(COLOR_RGB p1, double Thresh_md1);
#endif

```

```
#ifndef VICAL8
#define VICAL8
#include <vmemory.h>
typedef unsigned char byte;
unsigned char __far ** virtualloc(short xdim,short ydim);
#endif
```

¶

```

#include "comp_fnc.h"
int __cdecl compare_short_pnt ( const void *elem1,const void *elem2 )
{int i; double t1,t2;
 double a;
 t1=**(double**)elem1;t2=**(double**)elem2;
 a=t1-t2;
 i=(a?((a<0)?1:-1):0);
 return i;
}
//=====

int __cdecl compare_GOOD_DIR_LINE ( const void *elem1,const void
*elem2 )
{int i; double t1,t2;
 double a;
 t1=((GOOD_DIR_LINE*)elem1)->Qual;t2=((GOOD_DIR_LINE*)elem2)-
>Qual;
 a=t1-t2;
 i=(a?((a<0)?1:-1):0);
 return i;
}
//=====
=====
//=====
=====
int __cdecl compare_array_elem ( const void *elem1,const void *elem2 )
{int i;
 double a;
 a=**(double **)elem1-**(double **)elem2;
 i=(a?((a<0)?1:-1):0);
 return i;
}
//=====
=====

```

```

#include <stdio.h>
#include <stdlib.h>
#include <conio.h>
#include <float.h>
#include <graph.h>
#include "cor_fnc8.h"

//=====
=====
AVERAGE_VEC average(PCT tag,double Thr,
    COLOR_VEC (*p_funct)(COLOR_RGB p1,double Thresh_mdl))
{short r_t,c_t,n=0,i;
AVERAGE_VEC z={{0,0,0},{0,0,0}};
COLOR_VEC t_p;
for(r_t=0;r_t<tag.s_rows;r_t++)
for(c_t=0;c_t<tag.s_cols;c_t++)
{
n++;
t_p=p_funct(tag.get_pixel(r_t,c_t),Thr);
for(i=0;i<3;i++)
{
z.md[i]+=t_p.c[i];
z.var[i]+=t_p.c[i]*t_p.c[i];
}
}
double rr,nrev;
nrev=n?1.0/n:0;
for(i=0;i<3;i++)
{rr=(z.md[i]*nrev);
z.var[i]=z.var[i]*nrev;
}
return z;
}
//=====
=====
COLOR_VEC template_conv_1 (PCT target,PCT win,double
Thr,COLOR_VEC int_v,
    COLOR_VEC (*p_funct)(COLOR_RGB p1,double Thresh_mdl))
{short i,r,c,n;
COLOR_VEC w_p,t_p;
COLOR_VEC res={0,0,0};

for(n=r=0;
(r<target.s_rows)||((r<win.s_rows);r++)
for(c=0;
(c<target.s_cols)||((c<win.s_cols);c++)
{
n++;
w_p=p_funct(win.get_pixel(r,c),Thr);
t_p=p_funct(target.get_pixel(r,c),Thr);
for(i=0;i<3;i++)

```

```

        if(int_v.c[i])
            res.c[i]+=w_p.c[i]*t_p.c[i];
    }

double nrev;
nrev=n?1.0/n:0;
for(i=0;i<3;i++)
    res.c[i]=res.c[i]*nrev;
return res;
}
//=====
=====
COLOR_VEC template_abs_diff_1 (PCT target,PCT win,double
Thr,COLOR_VEC int_v,
    COLOR_VEC (*p_funct)(COLOR_RGB p1,double Thresh_mdl),
    AVERAGE_VEC t,AVERAGE_VEC w)
{short i,r,c,n;
COLOR_VEC w_p,t_p;
COLOR_VEC res={0,0,0},nor={0,0,0};

for(n=r=0;
    (r<target.s_rows)||((r<win.s_rows);r++)
for(c=0;
    (c<target.s_cols)||((c<win.s_cols);c++)
    {n++;
        w_p=p_funct(win.get_pixel(r,c),Thr);
        t_p=p_funct(target.get_pixel(r,c),Thr);
        for(i=0;i<3;i++)
            if(int_v.c[i])
                {res.c[i]+=fabs(w_p.c[i]-t_p.c[i]-w.md[i]+t.md[i]);
                    nor.c[i]+=__max (fabs(w_p.c[i]-w.md[i]),fabs(t_p.c[i]-t.md[i]));
                }
            // nor.c[i]+=2* fabs(w_p.c[i]-w.md[i]);
        }
        for(i=0;i<3;i++)
            if(int_v.c[i]&& nor.c[i])
                res.c[i]=1-res.c[i]/nor.c[i];
return res;
}
//=====
=====
COLOR_VEC Correlation(COLOR_VEC conv,AVERAGE_VEC
t,AVERAGE_VEC w,
    COLOR_VEC int_v)
{COLOR_VEC out;
int i;
double p,g;
for(i=0;i<3;i++)
    if(int_v.c[i])

```

```

        {p=(conv.c[i]-t.md[i]*w.md[i]);
        g=(t.var[i]-t.md[i]*t.md[i])*(w.var[i]-w.md[i]*w.md[i]);
        out.c[i]=g?p*p/g:0;
        out.c[i]=(p>0)?out.c[i]:-out.c[i];
        }
    else
        out.c[i]=0.0;
return out;
}
//=====
//=====
=====
/*COLOR_VEC Correlation_single_1(COLOR_VEC conv,AVERAGE_VEC
t,AVERAGE_VEC w,
    COLOR_VEC int_v)
{COLOR_VEC out;
int i;
double sngl_conv=0;
double sngl_var=0;
for(i=0;i<3;i++)
    if(int_v.c[i])
    {
        sngl_conv+=conv.c[i];
        sngl_var+=t.var[i]+w.var[i];
    }

out.c[0]=out.c[1]=out.c[2]=sngl_conv/sngl_var;

return out;
} */
//=====
=====
AVERAGE_VEC LineMoments(PCT &T, DIR_LINE Line,short map)
{COLOR_RGB col;
COLOR_VEC vc;
AVERAGE_VEC out={{0.0,0.0,0.0},{0.0,0.0,0.0}};
PT now;
short length=U_dist( Line);
double d;
short j,i,k,ST_X,END_X,now_X;
double relen,delta_V;
PT st, stop;
if(length<0)
{st=Line.End_p();
stop=Line.Start_p();
length=fabs(length);
}
else
{st=Line.Start_p(); stop=Line.End_p();

```



```

    }
    relen= length?1.0/length:0;
    ST_X=(short)st.u; END_X=(short)stop.u;

    if(delta_V=(stop.v-st.v)*relen)
        for(d=st.v+0.5,now_X=ST_X;now_X<END_X;now_X++,d+=delta_V)
        {
            // now.u=ST_X;now.v=d;
            // col=INTER_pix_color_rgb(T, now);
            col=T.get_pixel((short)d,now_X);
            switch (map)
            {
                case NTSC:;
                case New_plan:;
                case HSI:;
                case IHS:{
                    vc= PointColorFunct(map)(col,0);
                    for(k=0;k<3;k++)
                        out.md[k]+=vc.c[k];
                    break;
                }

                case RGB: {
                    out.md[0]+=col.r;//out.var[0]+=col.r*col.r;
                    out.md[1]+=col.g;//out.var[1]+=col.g*col.g;
                    out.md[2]+=col.b;//out.var[2]+=col.b*col.b;
                    break;
                }

                case LUMIN_THR:{
                    out.md[0]+=col.r+col.g+col.b;
                    break;
                }

            }

        };

    };

    for(j=0;j<3;j++)
    {
        out.md[j]*=relen;
        // out.var[j]/=length;
    }
    return out;
}
//=====
double Quality(AVERAGE_VEC * UpLineQuality)
{short i,j;
double out,out_even=0,out_odd=0,out2_even,out2_odd,sum=0;
for(j=0;j<3;j++)
{
    sum+=(UpLineQuality+0)->md[j]+(UpLineQuality+1)->md[j]+

```

```
        (UpLineQuality+2)->md[j]+(UpLineQuality+3)->md[j];
out2_even=(UpLineQuality+0)->md[j]+(UpLineQuality+1)->md[j]-
        (UpLineQuality+2)->md[j]-(UpLineQuality+3)->md[j];
out2_even*=out2_even;
out2_odd=(UpLineQuality+0)->md[j]-(UpLineQuality+1)->md[j]-
        (UpLineQuality+2)->md[j]+(UpLineQuality+3)->md[j];
out2_odd*=out2_odd;
out_even+=out2_even;
out_odd+=out2_odd;
    }
    out=(out_even+out_odd)*0.001;
    return out;
}
```

```

#include<stdlib.h>
#include<stdio.h>
#include<graph.h>
#include<math.h>
#include <io.h>
#include<fcntl.h>
#include <string.h>
#include <float.h>
#include <malloc.h>
#include "phdr.h"
#include "vical8.h"
#include "pic_mch8.h"

```

```
int cols, rows;
```

```
PHDR inhdr;
```

```

//=====
//=====
=====

```

```

void PCT::PutNextPix(COLOR_RGB p)
{
    *buffer_now++ = (unsigned char)p.r;
    *buffer_now++ = (unsigned char)p.g;
    *buffer_now++ = (unsigned char)p.b;
}

```

```

//=====
=====

```

```

int __cdecl compare_lum ( const void *elem1, const void *elem2 )
{int i; COLOR_RGB t1,t2;
 double a;
 t1=**(COLOR_RGB**)elem1;t2=**(COLOR_RGB**)elem2;
 a=(t1.r+t1.g+t1.b-t2.r-t2.g-t2.b);
 i=(a?((a<0)?1:-1):0);
 return i;
}

```

```

//=====
=====

```

```

//=====
=====

```

```

COLOR_RGB PCT::get_median_pixel(short y,short x,short neighbour)
{COLOR_RGB t={0,0,0},
 s[MEDIAN_AREA],*lum_order[MEDIAN_AREA];
 short n,i,j,xnow,ynow;
 int x3=3*x;
 unsigned char __far *buffer1;
 for(n=0,i=-neighbour;i<=neighbour;i++)

```

```

        for(j=-neighbour;j<=neighbour;j++)
            if(((ynow=y+i)<s_rows) && ((xnow=x+j)<s_cols))
            {
                buffer1= (unsigned char __far *) pict[ynow];
                t.r=*(buffer1+xnow*3+0);
                t.g=*(buffer1+xnow*3+1);
                t.b=*(buffer1+xnow*3+2);
                s[n]=t;
                *(lum_order+n)=s+n;
                n++;
            }
        qsort((void*)lum_order,n,sizeof(COLOR_RGB*),compare_lum );
        t=*lum_order[(n+1)>>1];
        return t;
    }
}

//=====
PCT::PCT( short n_cols, short n_rows)
{
    pict=
    (unsigned char __far **)virtualloc(3*n_cols,n_rows);
    if (!pict)
        {fprintf(stderr,"No memory for picture ");
        s_cols=0;
        s_rows=0;
        return;
        }
    buffer_now=pict[0];
    s_cols=n_cols;
    s_rows=n_rows;
}

//=====
void PCT::load_template(PCT source,SCR_PNT left_up_scr)
{
    short r_now, c_now;
    COLOR_RGB color_p;
    unsigned char *BUF,*B;
    BUF=(unsigned char *) malloc((size_t)
        (sizeof(unsigned char)*3*s_cols));
    if(!BUF) {fprintf(stderr," BUF ");exit(-1);}
    for (r_now=0; r_now<s_rows;r_now++)
        { for (B=BUF,c_now=0;c_now<s_cols;c_now++)
            {
                color_p=
                source.get_pixel(r_now+left_up_scr.r,c_now+left_up_scr.c);
                *B++=color_p.r;
                *B++=color_p.g;
                *B++=color_p.b;
            }
        }
}

```

```

        put_str(r_now,BUF);
    }

    free((void*)BUF);
}

//=====
int PCT::load_file_rgb(const char *name)
{ int er_r,short bytes;
  int y, infd;
  short ysize, xsize;
  char name_of_file[18];
  FILE *datfp;
  strcat(strcpy(name_of_file,name),".rgb");

  datfp=fopen(name_of_file,"rb");
  infd= _fileno( datfp );
  if(infd <= 0)
  { printf("bad name 1");
    return(1);
  }
  er_r=_read(infd, &inhdr, sizeof(PHDR));

  xsize=cols = inhdr.cols;
  ysize=rows = inhdr.rows;

  if(ysize > s_rows )
    ysize =s_rows;
  if(xsize > s_cols)
    xsize = s_cols;

  s_cols= xsize;
  s_rows=ysize;
  bytes =3 * xsize;

  byte Buf[2040];

  for(y = 0; y < ysize; y++) {
    read(infd, Buf, 3*xsize);
    put_str(y, (unsigned char*)Buf);
  }
  fclose(datfp);
  return 0;
}

//=====
PCT sign_storage_rgb(const char *name,struct _videoconfig vc)
{ int er_r;

```

```

PCT pict_now;
char name_of_file[40];
FILE *datfp;

strcat(strcpy(name_of_file,name),".rgb");

    int infd;
    short ysize, xsize;
    datfp=fopen(name_of_file,"rb");
    infd= _fileno( datfp );
    if(infd <= 0)
    {
        printf("bad name 1");
        return(pict_now);
    }
    er_r=_read(infd, &inhdr, sizeof(PHDR));
    if(er_r<0)
    {
        printf("bad name 1");
        return(pict_now);
    }
    fclose(datfp);
    xsize=cols = inhdr.cols;
    ysize=rows = inhdr.rows;
    if(ysize > vc.numypixels )
        ysize = vc.numypixels ;
    if(xsize > vc.numxpixels)
        xsize = vc.numxpixels;
    pict_now=PCT::PCT(xsize,ysize);
    if(pict_now.s_cols)
        pict_now.load_file_rgb(name);
    return(pict_now);
}

/*=====***/
void sign_present_RGB(PCT pict_scr,SCR_PNT left_up)
{short x,y,xsize,ysize;
  COLOR_RGB t;
  xsize=      pict_scr.s_cols;
  ysize=      pict_scr.s_rows;
  short c;
  for(y = 0; y < ysize; y++)
  {pict_scr.buffer_now=pict_scr.pict[y];
   for(x = 0; x < xsize; x++)
   {
       t=pict_scr.get_next();
       _setcolor(color_num (t.r>>2,t.g>>2,t.b>>2));
       _setpixel(x+left_up.c,y+left_up.r);
   }
  }
}

```

```

//=====
=====
const int stepr=9, stepg=9, stepb=21;

COLOR_RGB *make_palette()
{COLOR_RGB *plt=NULL;
 long now,Lut[256];
 long dr,dg,db;
 short i,j,k,num;

 for(i=0,db=0;i<4;i++,db+=stepb)
   for(j=0,dg=0;j<8;j++,dg+=stepg)
     for(k=0,dr=0;k<8;k++,dr+=stepr)
       { now=(db<<16)|(dg<<8)|dr;
         num=(i<<6)|(j<<3)|k;
         Lut[num]=now;
       }
   _remapallpalette((long __far *)Lut);
 return plt;
}
//=====
short color_num(short r,short g,short b)
{short num,i,j,k;
 i=(b+(stepb>>1))/stepb;
 j=(g+(stepg>>1))/stepg;
 k=(r+(stepr>>1))/stepr;
 num=(i<<6)|(j<<3)|k;
 return num;
}
//=====
void write_sign_rgb(char *name,PCT pict_now)
{ int er_r;;
 char name_of_file[80];
 strcat(strcpy(name_of_file,name),".rgb");
 FILE *datfp;
 int y, infd;
 datfp=fopen(name_of_file,"wb");
 infd= _fileno( datfp );
 if(infd <= 0)
   printf("bad name 1");

   inhdr.cols=pict_now.s_cols;
   inhdr.rows=pict_now.s_rows;
   inhdr.bpp=3;

   er_r=_write(infd, (void *)(&inhdr), sizeof(PHDR));
   for(y = 0; y < pict_now.s_rows; y++)
   {

```

```

#ifdef VIRTUAL
    void __far *buffer1;
    if ( (buffer1 = (void __far *)_vload( pict_now.pict[y],_VM_CLEAN )) ==
        NULL )
        { _vheapterm();exit( -1 );
        }
#else
    void *buffer1;
    buffer1= (void *) pict_now.pict[y];
#endif
    er_r=_write(infd, buffer1, sizeof(char)*pict_now.s_cols*3);
}
fclose(datfp);
strcat(strcpy(name_of_file,name),".sgn");
datfp=fopen(name_of_file,"w");
fprintf(datfp,"%s \n 0 0\n",name);
fprintf(datfp,"%d 0\n",pict_now.s_cols-1);
fprintf(datfp,"%d %d \n",pict_now.s_cols-1,pict_now.s_rows-1);
fprintf(datfp," 0 %d\n",pict_now.s_rows-1);
fclose(datfp);
}
//=====
=====

```



```

#include<stdlib.h>
#include<stdio.h>
#include<graph.h>
#include<math.h>
#include <io.h>
#include<fcntl.h>
#include <string.h>
#include <float.h>
//include <malloc.h>
//include "phdr.h"
//include "vical8.h"
#include "picture.h"

//int cols, rows;

//PHDR inhdr;
//=====
COLOR_RGB PCT::get_median_pixel(short y,short x,short neighbour)
{COLOR_RGB t={0,0,0},
  s[MEDIAN_AREA],*lum_order[MEDIAN_AREA];
  short n,i,j,xnow,ynow;
  int x3=3*x;
  unsigned char __far *buffer1;
  for(n=0,i=-neighbour;i<=neighbour;i++)
    for(j=-neighbour;j<=neighbour;j++)
      if(((ynow=y+i)<s_rows) && ((xnow=x+j)<s_cols))
        {
          buffer1= (unsigned char __far *) pict[ynow];
          t.r=(buffer1+xnow*3+0);
          t.g=(buffer1+xnow*3+1);
          t.b=(buffer1+xnow*3+2);
          s[n]=t;
          *(lum_order+n)=s+n;
          n++;
        }
  qsort((void*)lum_order,n,sizeof(COLOR_RGB*),compare_lum );
  t=*lum_order[(n+1)>>1];
  return t;
}
//=====
//=====
PCT::PCT( short n_cols, short n_rows)
{
  pict=
  (unsigned char __far **)virtalloc(3*n_cols,n_rows);
  if (!pict)
    {fprintf(stderr,"No memory for picture ");
     s_cols=0;
     s_rows=0;
    }
}

```

```

        return;
    }
    buffer_now=pict[0];
    s_cols=n_cols;
    s_rows=n_rows;
}

//=====================================================
void PCT::load_template(PCT source,SCR_PNT left_up_scr)
{
    short r_now, c_now;
    COLOR_RGB color_p;
    unsigned char *BUF,*B;
    BUF=(unsigned char *) malloc((size_t)
        (sizeof(unsigned char)*3*s_cols));
    if(!BUF) {fprintf(stderr," BUF ");exit(-1);}
    for (r_now=0; r_now<s_rows;r_now++)
        { for (B=BUF,c_now=0;c_now<s_cols; c_now++)
            {
                color_p=
                source.get_pixel(r_now+left_up_scr.r,c_now+left_up_scr.c);
                *B++=color_p.r;
                *B++=color_p.g;
                *B++=color_p.b;
            }
            put_str(r_now,BUF);
        }

        free((void*)BUF);
    }

//=====================================================
int PCT::load_file_rgb(const char *name)
{
    int er_r,short bytes;
    int y, infd;
    short ysize, xsize;
    char name_of_file[18];
    FILE *datfp;
    strcat(strcpy(name_of_file,name),".rgb");

    datfp=fopen(name_of_file,"rb");
    infd= _fileno( datfp );
    if(infd <= 0)
    {
        printf("bad name 1");
        return(1);
    }
    er_r=_read(infd, &inhdr, sizeof(PHDR));

    xsize=cols = inhdr.cols;
    ysize=rows = inhdr.rows;

```

```

    if(ysize > s_rows )
        ysize =s_rows;
        if(xsize > s_cols)
            xsize = s_cols;

    s_cols= xsize;
    s_rows=ysize;
    bytes =3 * xsize;

    byte Buf[2040];

    for(y = 0; y < ysize; y++) {
        read(infd, Buf, 3*cols);
        put_str(y, (unsigned char*)Buf);
    }
    fclose(datfp);
    return 0;
}

//=====
=====
PCT sign_storage_rgb(const char *name,struct _videoconfig vc)
{ int er_r;
  PCT pict_now;
  char name_of_file[40];
  FILE *datfp;

  strcat(strcpy(name_of_file,name),".rgb");

  int infd;
  short ysize, xsize;
  datfp=fopen(name_of_file,"rb");
  infd= _fileno( datfp );
  if(infd <= 0)
  {
      printf("bad name 1");
      return(pict_now);
  }
  er_r=_read(infd, &inhdr, sizeof(PHDR));
  if(er_r<0)
  {
      printf("bad name 1");
      return(pict_now);
  }
  fclose(datfp);
  xsize=cols = inhdr.cols;
  ysize=rows = inhdr.rows;
  if(ysize > vc.numypixels )
      ysize = vc.numypixels ;
  if(xsize > vc.numxpixels)

```

```

        xsize = vc.numxpixels;
    pict_now=PCT::PCT(xsize,ysize);
    if(pict_now.s_cols)
        pict_now.load_file_rgb(name);
    return(pict_now);
}

```

```

/*=====***/
void sign_present_RGB(PCT pict_scr,SCR_PNT left_up)
{short x,y,xsize,ysize;
  COLOR_RGB t;
  xsize=    pict_scr.s_cols;
  ysize=    pict_scr.s_rows;
  short c;
  for(y = 0; y < ysize; y++)
      {pict_scr.buffer_now=pict_scr.pict[y];
        for(x = 0; x < xsize; x++)
            {
                t=pict_scr.get_next();
                _setcolor(color_num (t.r>>2,t.g>>2,t.b>>2));
                _setpixel(x+left_up.c,y+left_up.r);
            }
        }
}

```

```

//=====
const int stepr=9, stepg=9, stepb=21;

COLOR_RGB *make_palette()
{COLOR_RGB *plt=NULL;
  long now,Lut[256];
  long dr,dg,db;
  short i,j,k,num;

  for(i=0,db=0;i<4;i++,db+=stepb)
      for(j=0,dg=0;j<8;j++,dg+=stepg)
          for(k=0,dr=0;k<8;k++,dr+=stepr)
              { now=(db<<16)|(dg<<8)|dr;
                num=(i<<6)|(j<<3)|k;
                Lut[num]=now;
              }
  _remapallpalette((long __far *)Lut);
  return plt;
}
//=====
short color_num(short r,short g,short b)
{short num,i,j,k;
  i=(b+(stepb>>1))/stepb;

```

```

j=(g+(stepg>>1))/stepg;
k=(r+(stepr>>1))/stepr;
num=(i<<6)|(j<<3)|k;
return num;
}
//=====
void write_sign_rgb(char *name,PCT pict_now)
{ int er_r;
  char name_of_file[80];
  strcat(strcpy(name_of_file,name),".rgb");
  FILE *datfp;
  int y, infd;
  datfp=fopen(name_of_file,"wb");
  infd= _fileno( datfp );
  if(infd <= 0)
    printf("bad name 1");

    inhdr.cols=pict_now.s_cols;
    inhdr.rows=pict_now.s_rows;
    inhdr.bpp=3;

    er_r=_write(infd, (void *)&inhdr, sizeof(PHDR));
    for(y = 0; y < pict_now.s_rows; y++)
    {
#ifdef VIRTUAL
      void __far *buffer1;
      if ( (buffer1 = (void __far *)_vload( pict_now.pict[y],_VM_CLEAN )) ==
NULL )
        { _vheapterm();exit( -1 );
        }
#else
      void *buffer1;
      buffer1= (void *) pict_now.pict[y];
#endif
      er_r=_write(infd, buffer1, sizeof(char)*pict_now.s_cols*3);
    }
  fclose(datfp);
  strcat(strcpy(name_of_file,name),".sgn");
  datfp=fopen(name_of_file,"w");
  fprintf(datfp,"%s \n 0 0\n",name);
  fprintf(datfp,"%d 0\n",pict_now.s_cols-1);
  fprintf(datfp,"%d %d \n",pict_now.s_cols-1,pict_now.s_rows-1);
  fprintf(datfp," 0 %d\n",pict_now.s_rows-1);
  fclose(datfp);
}
//=====
=====

```

```

#include <graph.h>
#include <stdlib.h>
#include <iostream.h>
#include "projctn8.h"
// For Constructor Calculate ratio of Lengthes.

// FOR CONSTRUCTOR Fill Members
//88888888888888
LINE_PROJECTION :: LINE_PROJECTION
(DIR_LINE l1, PT p, DIR_LINE pr, PT p_pr)
{ double P, Ro;
  RI=l1;
  Prj=pr;
  P = l1.RATIO_for_PT( p);
  Ro= pr.RATIO_for_PT( p_pr);
  if (P && (1-Ro))
    {proj_prm=(Ro-P)/(P*(1-Ro));return;};
  cout << "LINE UNRESOLVABLE\n";
}

PT cross_line(DIR_LINE Line1, DIR_LINE Line2)
{PT out;
 double det;
 if(det=Line1.a*Line2.b-Line2.a*Line1.b)
   {out.u=(Line1.b*Line2.c-Line1.c*Line2.b)/det;
    out.v=(Line1.c*Line2.a-Line2.c*Line1.a)/det;
   }
 else out.u=out.v=0;
#ifdef DEBUG
  _setcolor(0);
  _ellipse(_GFILLINTERIOR, (short) (out.u-1),(short) (out.v-1),
           (short) (out.u+1),(short) (out.v+1));
#endif
return out;
}

// Bield real space line connecting points X1 and X2
/// X1 belongs to line1, X2 to line2:
// PT pr1 - projection X1
// PT pr2 - projection X2
// LINE_PROJECTION line1, line2
DIR_LINE Line_for_PT_pr(LINE_PROJECTION line1, PT pr1,
                        LINE_PROJECTION line2, PT pr2)
{
  double P,Ro;
  PT p1, p2;

```

```

    Ro=line1.Prj.RATIO_for_PT( pr1);    //relative point position at line
    P=line1.P_for_Ro( Ro);
    p1=line1.Rl.PT_for_P(P);
    Ro=line2.Prj.RATIO_for_PT( pr2);    //relative point position at line
    P=line2.P_for_Ro( Ro);
    p2=line2.Rl.PT_for_P(P);
    DIR_LINE ln(p1,p2);
    return(ln);
}
// Bield projection line connecting points p1 and p2
/// p1 belongs to line1, p2 to line2:
// PT p1 - projection X1
// PT p2 - projection X2
// LINE_PROJECTION line1, line2

DIR_LINE Line_for_PT_rl(LINE_PROJECTION line1, PT p1,
                        LINE_PROJECTION line2, PT p2)

{double P,Ro;
  PT p1pr, p2pr;
  P=line1.Rl.RATIO_for_PT( p1);    //relative point position at line
  Ro=line1.Ro_for_P( P);
  p1pr=line1.Prj.PT_for_P(Ro);
  P=line2.Rl.RATIO_for_PT( p2);    //relative point position at line
  Ro=line2.Ro_for_P( P);
  p2pr=line2.Prj.PT_for_P(Ro);
  DIR_LINE ln(p1pr,p2pr);
  return(ln);
}
//=====
//=====
//=====
// =1 if point P in quagrange
int INside(DIR_LINE l1,DIR_LINE l2,DIR_LINE l3,DIR_LINE l4,int col,int row)
{return(l1.Left_Right(col,row)&& l2.Left_Right(col,row)&&
        l3.Left_Right(col,row)&& l4.Left_Right(col,row));
}
//=====

int U_dist(DIR_LINE l)
{return (l.end_pnt.u- l.st_pnt.u) ;
}
//=====

```

```

#include <stdlib.h>
#include <vmemory.h>
#include "res_mch8.h"
#define AVERAGE
#define SMOOSE_INT 1

extern double GAMMA;
//=====
=====
COLOR_VEC (*PointColorFunct(short M))(COLOR_RGB p1, double
Thresh_mdl)

{ switch ( M)
    {case NTSC:return(color_space_NTSC);
     case New_plan:return(color_space_NEW);
     case HSI:return(color_space_RGB);
     case RGB:return(color_space_RGB_simple);
     case LUMIN_THR:return(color_space_LUMIN_THR);
     case IHS:return(color_space_IHS);
    };
return NULL;
}

//=====
void GRAPH_OUT(int ex)
{
    _displaycursor( _GCURSORON );
    _setvideomode( _DEFAULTMODE );
    if (ex) exit (ex);
}
//=====
=====
int GRAPHICS_START(struct _videoconfig *p_vc,short GR_mode)
{
    _displaycursor( _GCURSOROFF );
    _setvideomode( GR_mode );
    if(_grstatus( ) <0) return(1);
    _getvideoconfig( p_vc );
    make_palette();
    return 0;
}
// -----sorting vertexes

int __cdecl c_comp( const void *elem1, const void *elem2)
{
    if (((SCR_PNT*)elem1)->c >((SCR_PNT *) elem2)->c)
        return 1;
    else
        if(((SCR_PNT *) elem1)->c < ((SCR_PNT *)elem2)->c)

```



```

        return -1;
    else return 0;
}
//=====
void match_vertex(SCR_PNT *v)
{ SCR_PNT vrt[4];
  int i;
  for (i=0;i<4;i++)
    vrt[i]=*(v+i);
  qsort((void *) vrt,4,sizeof(SCR_PNT),c_comp);
  if(vrt[0].r < vrt[1].r)
    {*(v+3)=vrt[0];*(v+2)=vrt[1];}
  else
    {*(v+3)=vrt[1];*(v+2)=vrt[0];}
  if(vrt[2].r < vrt[3].r)
    {*v=vrt[2];*(v+1)=vrt[3];}
  else
    {*v=vrt[3];*(v+1)=vrt[2];}
}
//=====
inline short interpol(short * s,double x,double y)
{double r=s[0]+x*y*(s[0]-s[1]+s[2]-s[3])+x*(s[1]-s[0])+y*(s[3]-s[0]);
  return((short)r);
}
//=====
=
// ONLY FOR INTER_pix_color_rgb
// and INTER_pix_color_rgb_median
//=====
=====
inline COLOR_RGB BE_linear(COLOR_RGB *p,double x_fract,double
y_fract)
    // loop over coolr
{short s[4];
  COLOR_RGB out_col;
  s[0]=p[0].r;
  s[1]=p[1].r;
  s[2]=p[2].r;
  s[3]=p[3].r;
  out_col.r=(short) interpol(s,x_fract,y_fract);
  s[0]=p[0].g;
  s[1]=p[1].g;
  s[2]=p[2].g;
  s[3]=p[3].g;
  out_col.g=(short) interpol(s,x_fract,y_fract);
  s[0]=p[0].b;
  s[1]=p[1].b;
  s[2]=p[2].b;
  s[3]=p[3].b;
}

```

```

    out_col.b=(short) interpol(s,x_fract,y_fract);
return out_col;
}
//=====
=====
COLOR_RGB INTER_pix_color_rgb(PCT p1, PT PT_now)
{
    int col0=(int)PT_now.u,
        row0=(int)PT_now.v;
    double x_fract=PT_now.u-col0, y_fract=PT_now.v-row0;
    COLOR_RGB p[4];
    p[0]=p1.get_pixel(row0,col0);
    p[1]=p1.get_next();
    p[3]=p1.get_pixel(row0+1,col0);
    p[2]=p1.get_next();

    return ( BE_linear( p, x_fract, y_fract));
}
//=====
COLOR_RGB INTER_pix_color_rgb_median(PCT p1, PT PT_now)
{
    int col0=(int)PT_now.u,
        row0=(int)PT_now.v;
    double x_fract=PT_now.u-col0, y_fract=PT_now.v-row0;
    COLOR_RGB p[4];
    p[0]=p1.get_median_pixel(row0,col0,SMOOSE_INT);
    p[1]=p1.get_median_pixel(row0,col0+1,SMOOSE_INT);
    p[2]=p1.get_median_pixel(row0+1,col0+1,SMOOSE_INT);
    p[3]=p1.get_median_pixel(row0+1,col0,SMOOSE_INT);

    return ( BE_linear( p, x_fract, y_fract));
}
//=====
#define NTSCr(c) ((c).r*0.6-(c).g*0.28-(c).b*0.32)
#define NTSCg(c) (0.21*(c).r-0.52*(c).g+0.31*(c).b)
#define NTSCw(c) (0.3*(c).r+0.59*(c).g+0.11*(c).b)
//=====
=
COLOR_VEC color_space_NTSC(COLOR_RGB p1, double Thresh_mdl)
{COLOR_VEC out={0,0,0};

    if((out.c[0]= NTSCw(p1))>Thresh_mdl)
    {out.c[1]= NTSCr(p1);
      out.c[2]= NTSCg(p1);
    }
    out.c[0]*=0.6;
return out;
}

```

77

```

//=====
=
COLOR_VEC color_space_NEW(COLOR_RGB p1, double Thresh_mdl)
{COLOR_VEC out={0,0,0};

    if((out.c[0]= NEWw(p1))>Thresh_mdl)
    {
        out.c[1]= NEWr(p1);
        out.c[2]= NEWg(p1);
//        out.c[1]= NEWr(p1)/out.c[0];
//        out.c[2]= NEWg(p1)/out.c[0];
    }
    out.c[0]*=0.6666666666666666;
return out;
}
//=====
==
#define LMNr(c) (c).r
#define LMNg(c) (c).g
#define LMNw(c) (((c).r+(c).g+(c).b)*0.333333333333)
#define Thresh_LUMINEN 136
//=====
==

COLOR_VEC color_space_LUMIN_THR(COLOR_RGB p1, double
Thresh_mdl)
{COLOR_VEC out={0,0,0};
double t;
    out.c[2]=out.c[1]=out.c[0]= ((t=LMNw(p1))>Thresh_mdl)?t:0;
return out;
}
//=====
==
const double SQ3= (sqrt(3.0));
const double PI05=(asin(1.0));
#define min3(c) (__min((c).r,__min((c).g,(c).b)))
#define max3(c) (__max((c).r,__max((c).g,(c).b)))
#define IHSh(c,d) (atan (SQ3*((c).g-(c).b)/d))
#define IHSi(c) (((c).r+(c).g+(c).b)*0.333333333333)
#define IHSs(c) ((double)(max3(c)-min3(c))/(double)(max3(c)+min3(c)))
//=====
==

COLOR_VEC color_space_IHS(COLOR_RGB p1, double Thresh_mdl)
{COLOR_VEC out={0,0,0};
double r=2*p1.r-p1.g-p1.b;

    out.c[1]= IHSi(p1);
    out.c[2]= out.c[1]?IHSs(p1):0;

```

```
    if(fabs(out.c[2])<Thresh_mdl)
        out.c[0]=0;
    else
        out.c[0]=r?IHSh(p1,r):(((p1.g-p1.b)<0)?-PI05:PI05);

    return out;
}
```

```
#include <stdlib.h>
#include <stdio.h>
#include <vmemory.h>
#include <malloc.h>

unsigned char __far ** virtalloc(short xdim, short ydim) {
    int y, j;
    unsigned char __far ** mem;

    mem = (unsigned char __far **) malloc( ydim * sizeof(unsigned char
    __far **));

    if(mem == NULL)
        return(NULL);

    for(y = 0; y < ydim; y++) {
        if ( (mem[y] = (unsigned char __far *) malloc(xdim*sizeof(char) )) ==
        NULL )
        {
            printf("NO MEMORY MAX size= %d", y);
            for(j=0; j<y; j++)
                free((void *)mem[j]);
            free((void *)mem);
            return(NULL);
        }
    }
    return(mem);
}
```

```

ORIGIN = PWB
ORIGIN_VER = 2.0
PROJ = LNS_CORR
PROJFILE = LNS_CORR.MAK
BUILDDIR = obj
DEBUG = 0

```

```

BRFLAGS = /o obj\$(PROJ).bsc
BSCMAKE = bscmake
SBRPACK = sbrpack
NMAKEBSC1 = set
NMAKEBSC2 = nmake
BROWSE = 1
CC = cl
CFLAGS_G = /W2 /BATCH /FR$*.sbr /Zn
CFLAGS_D = /f /Zi /Od
CFLAGS_R = /f- /Ot /Oi /Oj /Oe /Og /Gs
CXX = cl
CXXFLAGS_G = /AL /W4 /G2 /D_DOS /BATCH /FR$*.sbr /Zn
CXXFLAGS_D = /f- /Od /FPi87 /Zi /DMIC /DSINGLE_WIN
CXXFLAGS_R = /f- /Ot /Oj /Og /Oe /Oi /FPi87 /Gs /DMIC /DSINGLE_WIN
MAPFILE_D = NUL
MAPFILE_R = NUL
LFLAGS_G = /NOI /STACK:32000 /BATCH /ONERROR:NOEXE
LFLAGS_D = /CO /FAR /PACKC
LFLAGS_R = /EXE /FAR /PACKC
LINKER = link
ILINK = ilink
LRF = echo > NUL
ILFLAGS = /a /e
LLIBS_G = graphics lafxc
CVFLAGS = /25 /S
RUNFLAGS = ..\win4\S160_0 ..\win4\S160_ auto1

```

```

FILES = LNS_CORR.CPP ..\LIB\VICALLOC.CPP ..\LIB\PROJECTN7.CPP\
        ..\LIB\PIC_M7.CPP ..\LIB\RES_MCH7.CPP COR_FNC.CPP
COR_WIN.CPP
OBJS = obj\LNS_CORR.obj obj\VICALLOC.obj obj\PROJECTN7.obj
        obj\PIC_M7.obj
        obj\RES_MCH7.obj obj\COR_FNC.obj obj\COR_WIN.obj
SBRs = obj\LNS_CORR.sbr obj\VICALLOC.sbr obj\PROJECTN7.sbr
        obj\PIC_M7.sbr
        obj\RES_MCH7.sbr obj\COR_FNC.sbr obj\COR_WIN.sbr

```

```
all: obj\$(PROJ).exe
```

```

.SUFFIXES:
.SUFFIXES:
.SUFFIXES: .obj .sbr .cpp

```

```

obj\LNS_CORR.obj : LNS_CORR.CPP C:\C700\INCLUDE\stdlib.h\
C:\C700\INCLUDE\conio.h C:\C700\INCLUDE\stdio.h\
C:\C700\INCLUDE\string.h C:\C700\INCLUDE\graph.h\
C:\C700\INCLUDE\float.H C:\C700\INCLUDE\io.h
C:\C700\INCLUDE\time.h\
C:\C700\INCLUDE\ctype.h C:\C700\INCLUDE\iostream.h\
C:\C700\MFC\INCLUDE\afx.h LNS_CORR.h cor_win.h
c:\ilya\lib\vicalloc.h\
C:\C700\INCLUDE\ios.h C:\C700\INCLUDE\streamb.h\
C:\C700\INCLUDE\istream.h C:\C700\INCLUDE\ostream.h\
C:\C700\MFC\INCLUDE\afx.inl ..\LIB\projctn7.h ..\LIB\pic_mch7.h\
..\LIB\res_mch7.h c:\ilya\lib\lin_tm7.h C:\C700\INCLUDE\memory.h\
C:\C700\INCLUDE\math.h
!IF $(DEBUG)
    @$(CXX) @<<obj\$(PROJ).rsp
/c $(CXXFLAGS_G)
$(CXXFLAGS_D) /Foobj\LNS_CORR.obj LNS_CORR.CPP
<<
!ELSE
    @$(CXX) @<<obj\$(PROJ).rsp
/c $(CXXFLAGS_G)
$(CXXFLAGS_R) /Foobj\LNS_CORR.obj LNS_CORR.CPP
<<
!ENDIF

obj\LNS_CORR.sbr : LNS_CORR.CPP C:\C700\INCLUDE\stdlib.h\
C:\C700\INCLUDE\conio.h C:\C700\INCLUDE\stdio.h\
C:\C700\INCLUDE\string.h C:\C700\INCLUDE\graph.h\
C:\C700\INCLUDE\float.H C:\C700\INCLUDE\io.h
C:\C700\INCLUDE\time.h\
C:\C700\INCLUDE\ctype.h C:\C700\INCLUDE\iostream.h\
C:\C700\MFC\INCLUDE\afx.h LNS_CORR.h cor_win.h
c:\ilya\lib\vicalloc.h\
C:\C700\INCLUDE\ios.h C:\C700\INCLUDE\streamb.h\
C:\C700\INCLUDE\istream.h C:\C700\INCLUDE\ostream.h\
C:\C700\MFC\INCLUDE\afx.inl ..\LIB\projctn7.h ..\LIB\pic_mch7.h\
..\LIB\res_mch7.h c:\ilya\lib\lin_tm7.h C:\C700\INCLUDE\memory.h\
C:\C700\INCLUDE\math.h
!IF $(DEBUG)
    @$(CXX) @<<obj\$(PROJ).rsp
/Zs $(CXXFLAGS_G)
$(CXXFLAGS_D) /FRobj\LNS_CORR.sbr LNS_CORR.CPP
<<
!ELSE
    @$(CXX) @<<obj\$(PROJ).rsp
/Zs $(CXXFLAGS_G)
$(CXXFLAGS_R) /FRobj\LNS_CORR.sbr LNS_CORR.CPP
<<

```



!ENDIF

obj\VICALLOC.obj : ..\LIB\VICALLOC.CPP C:\C700\INCLUDE\stdlib.h  
 C:\C700\INCLUDE\stdio.h C:\C700\INCLUDE\vmemory.h\  
 C:\C700\INCLUDE\malloc.h

!IF \$(DEBUG)

@\$(CXX) @<<obj\\$(PROJ).rsp

/c \$(CXXFLAGS\_G)

\$(CXXFLAGS\_D) /Foobj\VICALLOC.obj ..\LIB\VICALLOC.CPP

<<

!ELSE

@\$(CXX) @<<obj\\$(PROJ).rsp

/c \$(CXXFLAGS\_G)

\$(CXXFLAGS\_R) /Foobj\VICALLOC.obj ..\LIB\VICALLOC.CPP

<<

!ENDIF

obj\VICALLOC.sbr : ..\LIB\VICALLOC.CPP C:\C700\INCLUDE\stdlib.h  
 C:\C700\INCLUDE\stdio.h C:\C700\INCLUDE\vmemory.h\  
 C:\C700\INCLUDE\malloc.h

!IF \$(DEBUG)

@\$(CXX) @<<obj\\$(PROJ).rsp

/Zs \$(CXXFLAGS\_G)

\$(CXXFLAGS\_D) /FRobj\VICALLOC.sbr ..\LIB\VICALLOC.CPP

<<

!ELSE

@\$(CXX) @<<obj\\$(PROJ).rsp

/Zs \$(CXXFLAGS\_G)

\$(CXXFLAGS\_R) /FRobj\VICALLOC.sbr ..\LIB\VICALLOC.CPP

<<

!ENDIF

obj\PROJECTN7.obj : ..\LIB\PROJECTN7.CPP C:\C700\INCLUDE\graph.h  
 C:\C700\INCLUDE\stdlib.h C:\C700\INCLUDE\iostream.h  
 ..\LIB\projctn7.h\  
 C:\C700\INCLUDE\ios.h C:\C700\INCLUDE\streamb.h\  
 C:\C700\INCLUDE\istream.h C:\C700\INCLUDE\ostream.h\  
 C:\C700\INCLUDE\math.h

!IF \$(DEBUG)

@\$(CXX) @<<obj\\$(PROJ).rsp

/c \$(CXXFLAGS\_G)

\$(CXXFLAGS\_D) /Foobj\PROJECTN7.obj ..\LIB\PROJECTN7.CPP

<<

!ELSE

@\$(CXX) @<<obj\\$(PROJ).rsp

/c \$(CXXFLAGS\_G)

\$(CXXFLAGS\_R) /Foobj\PROJECTN7.obj ..\LIB\PROJECTN7.CPP

<<

!ENDIF

```

obj\PROJECTN7.sbr : ..\LIB\PROJECTN7.CPP C:\C700\INCLUDE\graph.h\
C:\C700\INCLUDE\stdlib.h C:\C700\INCLUDE\iostream.h
..\LIB\projctn7.h\
C:\C700\INCLUDE\ios.h C:\C700\INCLUDE\streamb.h\
C:\C700\INCLUDE\istream.h C:\C700\INCLUDE\ostream.h\
C:\C700\INCLUDE\math.h
!IF $(DEBUG)
    @$(CXX) @<<obj\$(PROJ).rsp
/Js $(CXXFLAGS_G)
$(CXXFLAGS_D) /Fobj\PROJECTN7.sbr ..\LIB\PROJECTN7.CPP
<<
!ELSE
    @$(CXX) @<<obj\$(PROJ).rsp
/Js $(CXXFLAGS_G)
$(CXXFLAGS_R) /Fobj\PROJECTN7.sbr ..\LIB\PROJECTN7.CPP
<<
!ENDIF

obj\PIC_M7.obj : ..\LIB\PIC_M7.CPP C:\C700\INCLUDE\stdlib.h\
C:\C700\INCLUDE\stdio.h C:\C700\INCLUDE\graph.h
C:\C700\INCLUDE\math.h\
C:\C700\INCLUDE\io.h C:\C700\INCLUDE\fcntl.h
C:\C700\INCLUDE\string.h\
C:\C700\INCLUDE\float.H C:\C700\INCLUDE\malloc.h ..\LIB\phdr.h\
c:\ilya\lib\vicalloc.h ..\LIB\pic_mch7.h C:\C700\INCLUDE\vmemory.h\
..\LIB\projctn7.h
!IF $(DEBUG)
    @$(CXX) @<<obj\$(PROJ).rsp
/c $(CXXFLAGS_G)
$(CXXFLAGS_D) /Foobj\PIC_M7.obj ..\LIB\PIC_M7.CPP
<<
!ELSE
    @$(CXX) @<<obj\$(PROJ).rsp
/c $(CXXFLAGS_G)
$(CXXFLAGS_R) /Foobj\PIC_M7.obj ..\LIB\PIC_M7.CPP
<<
!ENDIF

obj\PIC_M7.sbr : ..\LIB\PIC_M7.CPP C:\C700\INCLUDE\stdlib.h\
C:\C700\INCLUDE\stdio.h C:\C700\INCLUDE\graph.h
C:\C700\INCLUDE\math.h\
C:\C700\INCLUDE\io.h C:\C700\INCLUDE\fcntl.h
C:\C700\INCLUDE\string.h\
C:\C700\INCLUDE\float.H C:\C700\INCLUDE\malloc.h ..\LIB\phdr.h\
c:\ilya\lib\vicalloc.h ..\LIB\pic_mch7.h C:\C700\INCLUDE\vmemory.h\
..\LIB\projctn7.h
!IF $(DEBUG)
    @$(CXX) @<<obj\$(PROJ).rsp

```

```

/Zs $(CXXFLAGS_G)
$(CXXFLAGS_D) /FRobj\PIC_M7.sbr ..\LIB\PIC_M7.CPP
<<
!ELSE
    @$(CXX) @<<obj\$(PROJ).rsp
/Zs $(CXXFLAGS_G)
$(CXXFLAGS_R) /FRobj\PIC_M7.sbr ..\LIB\PIC_M7.CPP
<<
!ENDIF

obj\RES_MCH7.obj : ..\LIB\RES_MCH7.CPP C:\C700\INCLUDE\stdlib.h\
C:\C700\INCLUDE\vmemory.h ..\LIB\pic_mch7.h ..\LIB\res_mch7.h\
C:\C700\INCLUDE\graph.h ..\LIB\projctn7.h C:\C700\INCLUDE\math.h
!IF $(DEBUG)
    @$(CXX) @<<obj\$(PROJ).rsp
/c $(CXXFLAGS_G)
$(CXXFLAGS_D) /Foobj\RES_MCH7.obj ..\LIB\RES_MCH7.CPP
<<
!ELSE
    @$(CXX) @<<obj\$(PROJ).rsp
/c $(CXXFLAGS_G)
$(CXXFLAGS_R) /Foobj\RES_MCH7.obj ..\LIB\RES_MCH7.CPP
<<
!ENDIF

obj\RES_MCH7.sbr : ..\LIB\RES_MCH7.CPP C:\C700\INCLUDE\stdlib.h\
C:\C700\INCLUDE\vmemory.h ..\LIB\pic_mch7.h ..\LIB\res_mch7.h\
C:\C700\INCLUDE\graph.h ..\LIB\projctn7.h C:\C700\INCLUDE\math.h
!IF $(DEBUG)
    @$(CXX) @<<obj\$(PROJ).rsp
/Zs $(CXXFLAGS_G)
$(CXXFLAGS_D) /FRobj\RES_MCH7.sbr ..\LIB\RES_MCH7.CPP
<<
!ELSE
    @$(CXX) @<<obj\$(PROJ).rsp
/Zs $(CXXFLAGS_G)
$(CXXFLAGS_R) /FRobj\RES_MCH7.sbr ..\LIB\RES_MCH7.CPP
<<
!ENDIF

obj\COR_FNC.obj : COR_FNC.CPP C:\C700\INCLUDE\stdio.h
C:\C700\INCLUDE\stdlib.h\
C:\C700\INCLUDE\conio.h C:\C700\INCLUDE\float.H\
C:\C700\INCLUDE\graph.h cor_fnc.h ..\LIB\pic_mch7.h
..\LIB\res_mch7.h\
C:\C700\INCLUDE\vmemory.h ..\LIB\projctn7.h
C:\C700\INCLUDE\math.h
!IF $(DEBUG)
    @$(CXX) @<<obj\$(PROJ).rsp

```

```

/c $(CXXFLAGS_G)
$(CXXFLAGS_D) /Foobj\COR_FNC.obj COR_FNC.CPP
<<
!ELSE
    @$(CXX) @<<obj\$(PROJ).rsp
/c $(CXXFLAGS_G)
$(CXXFLAGS_R) /Foobj\COR_FNC.obj COR_FNC.CPP
<<
!ENDIF

obj\COR_FNC.sbr : COR_FNC.CPP C:\C700\INCLUDE\stdio.h
C:\C700\INCLUDE\stdlib.h\
    C:\C700\INCLUDE\conio.h C:\C700\INCLUDE\float.H\
    C:\C700\INCLUDE\graph.h cor_fnc.h ..\LIB\pic_mch7.h
..\LIB\res_mch7.h\
    C:\C700\INCLUDE\vmemory.h ..\LIB\projctn7.h
C:\C700\INCLUDE\math.h
!IF $(DEBUG)
    @$(CXX) @<<obj\$(PROJ).rsp
/Zs $(CXXFLAGS_G)
$(CXXFLAGS_D) /FRobj\COR_FNC.sbr COR_FNC.CPP
<<
!ELSE
    @$(CXX) @<<obj\$(PROJ).rsp
/Zs $(CXXFLAGS_G)
$(CXXFLAGS_R) /FRobj\COR_FNC.sbr COR_FNC.CPP
<<
!ENDIF

obj\COR_WIN.obj : COR_WIN.CPP C:\C700\INCLUDE\vmemory.h\
C:\C700\INCLUDE\stdio.h C:\C700\INCLUDE\stdlib.h\
C:\C700\INCLUDE\conio.h C:\C700\INCLUDE\malloc.h\
C:\C700\INCLUDE\float.H C:\C700\INCLUDE\graph.h\
C:\C700\INCLUDE\string.h LNS_CORR.h cor_fnc.h ..\LIB\projctn7.h\
..\LIB\pic_mch7.h ..\LIB\res_mch7.h c:\ilya\lib\lin_tm7.h\
C:\C700\INCLUDE\math.h
!IF $(DEBUG)
    @$(CXX) @<<obj\$(PROJ).rsp
/c $(CXXFLAGS_G)
$(CXXFLAGS_D) /Foobj\COR_WIN.obj COR_WIN.CPP
<<
!ELSE
    @$(CXX) @<<obj\$(PROJ).rsp
/c $(CXXFLAGS_G)
$(CXXFLAGS_R) /Foobj\COR_WIN.obj COR_WIN.CPP
<<
!ENDIF

obj\COR_WIN.sbr : COR_WIN.CPP C:\C700\INCLUDE\vmemory.h\

```

```

C:\C700\INCLUDE\stdio.h C:\C700\INCLUDE\stdlib.h
C:\C700\INCLUDE\conio.h C:\C700\INCLUDE\malloc.h
C:\C700\INCLUDE\float.H C:\C700\INCLUDE\graph.h\
C:\C700\INCLUDE\string.h LNS_CORR.h cor_fnc.h ..\LIB\projctn7.h\
..\LIB\pic_mch7.h ..\LIB\res_mch7.h c:\ilya\lib\lin_tm7.h\
C:\C700\INCLUDE\math.h
!IF $(DEBUG)
    @$(CXX) @<<obj$(PROJ).rsp
/Zs $(CXXFLAGS_G)
$(CXXFLAGS_D) /Fobj\COR_WIN.sbr COR_WIN.CPP
<<
!ELSE
    @$(CXX) @<<obj$(PROJ).rsp
/Zs $(CXXFLAGS_G)
$(CXXFLAGS_R) /Fobj\COR_WIN.sbr COR_WIN.CPP
<<
!ENDIF

obj$(PROJ).bsc : $(SBRs)
    $(BSCMAKE) @<<
$(BRFLAGS) $(SBRs)
<<

obj$(PROJ).exe : $(OBJS)
    -$(NMAKEBSC1) MAKEFLAGS=
    -$(NMAKEBSC2) $(NMFLAGS) -f $(PROJFILE) obj$(PROJ).bsc
!IF $(DEBUG)
    $(LRF) @<<obj$(PROJ).lrf
$(RT_OBJS: = +^
) $(OBJS: = +^
)
$@
$(MAPFILE_D)
$(LIBS: = +^
) +
$(LLIBS_G: = +^
) +
$(LLIBS_D: = +^
)
$(DEF_FILE) $(LFLAGS_G) $(LFLAGS_D);
<<
!ELSE
    $(LRF) @<<obj$(PROJ).lrf
$(RT_OBJS: = +^
) $(OBJS: = +^
)
$@
$(MAPFILE_R)

```

```

$(LIBS: = +^
) +
$(LLIBS_G: = +^
) +
$(LLIBS_R: = +^
)
$(DEF_FILE) $(LFLAGS_G) $(LFLAGS_R);
<<
!ENDIF
    $(LINKER) @obj\$(PROJ).lrf

```

```

.cpp.obj:
!IF $(DEBUG)
    @$(CXX) @<<obj\$(PROJ).rsp
/c $(CXXFLAGS_G)
$(CXXFLAGS_D) /Fo$@ $<
<<
!ELSE
    @$(CXX) @<<obj\$(PROJ).rsp
/c $(CXXFLAGS_G)
$(CXXFLAGS_R) /Fo$@ $<
<<
!ENDIF

```

```

.cpp.sbr:
!IF $(DEBUG)
    @$(CXX) @<<obj\$(PROJ).rsp
/Zs $(CXXFLAGS_G)
$(CXXFLAGS_D) /FR$@ $<
<<
!ELSE
    @$(CXX) @<<obj\$(PROJ).rsp
/Zs $(CXXFLAGS_G)
$(CXXFLAGS_R) /FR$@ $<
<<
!ENDIF

```

```

run: obj\$(PROJ).exe
    obj\$(PROJ).exe $(RUNFLAGS)

```

```

debug: obj\$(PROJ).exe
    CV $(CVFLAGS) obj\$(PROJ).exe $(RUNFLAGS)

```

```
void corr_win_proto(PCT win_source, PCT Proto, SCR_PNT win_size  
    , short win_step, double CORR_THRESH,  
    short *StripEnds) ;
```

```

//      Module calculates correlation functions of PROTO_1 and set of
// prototypes. Set of prototypes' names is defined by a MASK correspondes
// to names generated by MAKEPRB and has next structure:
//      [path]&RRW_P.rgb
// Where
//      [path] - optional name of directory;
//      &      - first letter of file name
//      RR      - two digits corresponding to prototype's hight
//              (RR= 16| 32 | 48 | 64)
//      W      - number corresponding to window number (see
MAKEPRB
//              description.
//      P      prototype Number
// MASK includes ONLY [path]&RRW_      and programme will
// calculate correlation functions for prototypes with P from 0 to
// first not existing number.

// COMMAND STRING
//
// Ins_corr <PROTO_1_Name> <MASK> [CommandFile]
//
//      <PROTO_1_Name>      File name of PROTOTYPE without
extention
//      <MASK>      Mask for prototypes FileNames without extention
and
//      Prototype's number.
//      [CommandFile]      Optional ASCII file with a run time parameters.
//
// INPUT
//      RGB files of prototypes and corresponding .SGN files created by
// module MAKEPRB.
// RUN TIME parameters:
//
//      0 0 0 0      -shift for all cases have to be 0
//      <CalorSpace>
//      We have used 1 - as worked only with a luminance
//      <Window width>
//      We have used 8
//SEE ALSO FILE "LNS_CORR.INI"
// OUTPUT
//      Correlation functions in PROTO_1.DBC file.

#include <stdlib.h>
#include <conio.h>
#include <stdio.h>
#include <string.h>
#include <graph.h>
#include <float.H>

```



```
CString PROTOCOL_NAME;  
CString PROTOCOL_START;  
CString PROTO1_HEADER=CString::CString(
```



```

#include <fstream.h>
int ReadStrInf(char *name,short *StD)
{ifstream InpF;
 char a[80];
 strcat(strcpy(a,name),".str");
 short i;
 InpF.open(a,ios::in|ios::nocreate);
 if(InpF.fail())
 {InpF.clear(0);
  return 1;
 }
 InpF.getline(a,80,'\n');
 InpF.getline(a,80,'\n');
 InpF.getline(a,80,'\n');
 InpF.getline(a,80,'\n');
 InpF.getline(a,80,'\n');
 InpF.getline(a,80,'\n');
 i=0;
 do
 {
  InpF>>StD[i++];
  if (InpF.eof())|| i>17
  { StD[-i]=-1;
   break;
  }
  InpF>>StD[i++];
 }
 while(1);
 InpF.close();
 return 0;
}

```

```

//=====
=====
SCR_PNT winsize;
//=====
=====
int main(int argc,char* argv[])
{int FLG_F=0, FLG_WRIGHT=0;
 FILE *datainf;
 short winstep, map_std;
 short n=0;
 SCR_PNT t_pos;
 if((argc != 3) && (argc !=4))
 {
  printf(" target-file proto_file_mask \n");
  FLG_F=0;
  return(1);
 }

```

```

        else
            if(argc ==4)
            {
                FLG_F=1;
                if(!(datainf=fopen(argv[3],"r"))) return 0;
            }
        if(FLG_F)
        {
            get_shift_f(datainf,sh); // GET SCALE
            get_number_3_f(datainf); // GET SCALE
        }
        else
        {
            get_shift(sh); // GET SCALE 0
            get_number_3();
        }

        strcpy(ARGV_1,argv[1]);
        PROTOCOL_NAME=argv[1];
        PROTOCOL_NAME+=".dbc";
        init_protocol(argv[1]);

        // ===== GRAPHICS START
            if(GRAPHICS_START(&vc,GRAPHMODE)) exit(-1);
        //===== TARGET PICTURE name and vertexes
        SCR_PNT target_pos; // CONSTRUCTOR default 0,0
        short StrDescr[17];
        _splitpath( argv[1], drive,dir,TAG_name,ext );
        pict_target=sign_storage_rgb(argv[1],vc );
        if(ReadStrInf(argv[1],StrDescr))
        {
            printf("STR PROTO not exist"); GRAPH_OUT(-1);return -1;
        };

        winsize.r=pict_target.s_rows;
        winstep=winsize.c;
        // "PROTO_File\FRAME_Name\STRING_name\S_name\SLength\WinLengt
        h\SPACE\n";
        fprintf(PROTOCOL,"%s %8s\t%6s\t%12s\t%4d\t%4d\t%12s\n%s",
            (const char
            *)PROTOCOL_START,FRAME_Name,STRING_name,SIGN_name,
            pict_target.s_cols,winsize.c,SP[MAP],
            (const char *) PROTO_TAG_HEADER);

        //===== PROTOTYPE LOOP OVER names
        char proto_name[NAME_LENGTH],buff[4];
        SCR_PNT proto_pos,z;
        //loop over masks
        //return file name without extension in "name" and TRUE 1 if file exist;
        short proto_number=0; // # 0;
        while( next_pict( proto_name,argv[2],".rgb", proto_number))
        {
            proto_number=-1; //next;
        }
        SCR_PNT pr_v[4];

```

```

// PROTO INFORMATION IN PROTOCOL
    pict_proto=sign_storage_rgb(proto_name,vc );
    picture_inf_num_2(proto_name,pr_v,0,".str"); //only for SIGN_name
// "TAG_File\tFRAME_Name\tSTRING_name\tS_Name\tLegnth\n"
    fprintf(PROTOCOL," %12s\t %8s\t %6s\t %12s\t%4d\n",

proto_name,FRAME_Name,STRING_name,SIGN_name,pict_proto.s_cols);

    TAG_hight=pict_proto.s_rows;
// TARGET PRESENTATION
    _clearscreen(_GCLEARSCREEN);
    proto_pos.c=target_pos.c=10;
    proto_pos.r=(target_pos.r=10)+pict_target.s_rows+5;
    sign_present_RGB( pict_target,target_pos);
    sign_present_RGB(pict_proto,proto_pos);
//=====
    corr_win_proto(pict_target, pict_proto,
        winsize, winstep,CORR_THRESH,StrDescr);

    pict_proto.free_PCT();
}
_displaycursor( _G_CURSORON );
_setvideomode( _DEFAULTMODE );
pict_target.free_PCT();
fclose(PROTOCOL);
return(0);
}

//
=====
=====
void get_shift_f(FILE *f,double * sh) // INITIALISATION GRAPHICMODE,
GET SCALE
{int i;
    for(i=0;i<4; i++)
    {
        fscanf(f,"%lf %lf\n",sh+i++,sh+i);
    }
}

//
=====
=====
void get_shift(double * sh) // INITIALISATION GRAPHICMODE, GET SCALE
{int i;
    cout<< "vertexes shift over rows ( top_right, bottom_right, bottom_left,
top_left \n";
    for (i=0;i<4; i++)
        cin>>sh[i];
}

```

```

//=====
int get_number_3() // INITIALISATION GRAPHICMODE, GET SCALE
{int R;
    _displaycursor( _G_CURSORON );
    _setvideomode( _DEFAULTMODE );
    cout << " [<0 EXIT], color_map (0-NTSC, 1-HSI,2-NEW,3-RGB,4-
LUMIN_THR 5-HSI\n";
    cout<<"WIN_SIZE\n ";
    cin >>MAP>>winsize.c;
    _displaycursor( _G_CURSOROFF );
    _setvideomode( GRAPHMODE );
    make_palette();
return R;
}
//=====
int get_number_3_f(FILE *f) // INITIALISATION GRAPHICMODE, GET
SCALE
{int R;
    fscanf(f," %d %d ",&MAP, &(winsize.c));
return 1;
}
//=====
int picture_inf(char *name,SCR_PNT *vertexes)
{int i;
    char new_name[25];
    FILE *datfp;
    strcat(strcpy(new_name,name),".sgn");
    if(!(datfp=fopen(new_name,"r"))) return 0;
    fscanf(datfp,"%s\n",new_name);
    for(i=0;i<4;i++)
        fscanf(datfp,"%d %d\n",&(vertexes[i].c),&(vertexes[i].r));
    fclose(datfp);
return 1;
}
//=====
//=====
int picture_inf_num_2(char *name,SCR_PNT *vertexes,short n,char
*ext=".sgn")
{int i,j;
    char new_name[45];
    FILE *datfp;
    strcat(strcpy(new_name,name),ext);

    if(!(datfp=fopen(new_name,"r"))) return 0;
    fscanf(datfp,"%s %s %s %s %s\n");
    fscanf(datfp,"%s %s %s %s %s",&f_name,&FILE_name,&FRAME_name,
&STRING_name,&SIGN_name);

    for(j=0;j<n+1;j++)

```

```

        for(i=0;i<4;i++)
            if(fscanf(datfp,"%d %d\n",&(vertexes[i].c),&(vertexes[i].r))==EOF)
                {fclose (datfp); return 0;}
        fclose(datfp);
    return 1;
}
//=====
/*void write_sign_inf(char *pr,PCT pict_now)
{ char fl_fp[50],f_name[9];

    int FLG;
    FILE *dathere,*database;
    _splitpath( pr, drive,dir,f_name,ext );
    strcat(strcpy(fl_fp,pr),".sgn");
    dathere=fopen(fl_fp,"w");
    FLG=_access("PROTODB.1", 0 );// -1 if not exist

    if(!(database=fopen("PROTODB.1","a")))
        {strcpy(fl_fp,"CAN NOT CREATE D_BASE FILE");
        exit(-1);}
    fprintf(dathere, " WIN_name FILE_name FRAME_Name STRING_name
SIGN_name\n ");
    fprintf(dathere,"%8s %9s %10s %11s %9s \n",f_name, FILE_name,

FRAME_Name,STRING_name,SIGN_name);
    if(FLG)
        fprintf(database, " WIN_name FILE_name FRAME_Name STRING_name
SIGN_name\n ");
        fprintf(database,"%8s %9s %10s %11s %9s \n",f_name, FILE_name,

FRAME_Name,STRING_name,SIGN_name);
    fprintf(dathere,"%d 0\n",pict_now.s_cols-1);
    fprintf(dathere,"%d %d \n",pict_now.s_cols-1,pict_now.s_rows-1);
    fprintf(dathere," 0 %d\n",pict_now.s_rows-1);
    fprintf(dathere," 0 0\n");

    fclose(dathere);
    fclose(database);
} */
//=====
//=====
int picture_inf_num(char *name,SCR_PNT *vertexes,short n)
{int i,j;
    char new_name[25];
    FILE *datfp;
    strcat(strcpy(new_name,name),".sgn");
    if(!(datfp=fopen(new_name,"r"))) return 0;
    fscanf(datfp,"%s\n",new_name);

```

```

    for(j=0;j<n+1;j++)
        for(i=0;i<4;i++)
            if(fscanf(datfp,"%d %d\n",&(vertexes[i].c),&(vertexes[i].r))==EOF)
                {fclose (datfp); return 0;}
    fclose(datfp);
    return 1;
}
//=====================================================
int picture_inf_num_new(char *name,SCR_PNT *vertexes,short n)
{int i,j;
 char new_str[80];
 FILE *datfp;
 int r,FLG=0;
 strcat(strcpy(new_str,name),".sgn");
 if(!(datfp=fopen(new_str,"r"))) return 0;
 r=fscanf(datfp,"%[^\n]s ",new_str);
 r=fscanf(datfp," %[^\n]s ",new_str);
 if(__iscsymf( (int)new_str[0]))//FILE INFORMATION )
    //(letter or underscore)
    {sscanf(new_str," %s %s %s %s",&FILE_name,&FRAME_Name,
        &STRING_name,&SIGN_name);
    r=fscanf(datfp," %[^\n]s ",new_str);
    }
    for(j=0;j<n+1;j++)
        for(i=0;i<4;i++)
            {if(FLG)
                if(fscanf(datfp," %[^\n]s",new_str)==EOF)
                    {fclose (datfp); return 0;}
                FLG=1;
                sscanf(new_str," %d %d",&(vertexes[i].c),&(vertexes[i].r));
            }
    fclose(datfp);
    return 1;
}
//=====================================================

```



```

    }
//-----
/*COLOR_VEC int_value_1(PCT w,double Thr,
    COLOR_VEC (*p_func)(COLOR_RGB p1,double
    Thresh_md1),AVERAGE_VEC w_av)
{COLOR_VEC col,sum[9][9],out,c1;
const COLOR_VEC z={0,0,0};

    short h_t,v_t,i,x,y,h,v,
    half_x=w.s_cols>>1,half_y=w.s_rows>>1,
    quot_x=w.s_cols>>2,quot_y=w.s_rows>>2;
    long n;

    for(h=0;h<HOR_HARM+1;h++)
        for(v=0;v<VERT_HARM+1;v++)
            sum[v][h].c[0]=sum[v][h].c[1]=sum[v][h].c[2]=0.0;
    n=w.s_cols*w.s_rows;
    n*=n;
    for(y=0;y<w.s_rows;y++)
        for(v=0;v<VERT_HARM+1;v++)
            {
                v_t=y*((v+1)>>1);
                v_t=(v_t+(v & 0x0001 ? quot_y:0))/half_y;
                v_t &= 0x0001;
                for(x=0;x<w.s_cols;x++)
                    {col=p_func(w.get_pixel(y,x),Thr);
                        c1=sign_for_col(v_t,col);
                        for(h=0;h<HOR_HARM+1;h++)
                            {
                                h_t=x*((h+1)>>1);
                                h_t=(h_t+(h & 0x0001 ? quot_x:0))/half_x;
                                h_t &= 0x0001;
                                c1=sign_for_col(h_t,c1);
                                for(i=0;i<3;i++)
                                    sum[v][h].c[i]+=c1.c[i];
                            }
                        }
            }
    }
double s0,dd,max_v=0,th;
for(dd=i=0;i<3;i++)
    {for(s0=h=0;h<HOR_HARM+1;h++)
        for(v=0;v<VERT_HARM+1;v++)
            if(h||v)
                s0+=sum[v][h].c[i]*sum[v][h].c[i];
        s0/=n;
        dd=w_av.var[i]+w_av.md[i]*w_av.md[i];
        out.c[i]=(dd?s0/dd:1);
        max_v=(max_v<out.c[i]?out.c[i]:max_v;
    }
}

```

```

    for(i=0;i<3;i++)
        {th=out.c[i]/max_v;
        //          THRESHOLDING
        if(th<0.2)
            out.c[i]=0;
        }
    return out;
} */
//=====
//=====
COLOR_VEC (*PointColFunct())(COLOR_RGB p1,double Thresh_mdl)

{ switch ( MAP)
    {case NTSC:return(color_space_NTSC);
    case New_plan:return(color_space_NEW);
    case HSI:return(color_space_RGB);
    case RGB:return(color_space_RGB_simple);
    case LUMIN_THR:return(color_space_LUMIN_THR);
    case IHS:return(color_space_IHS);
    };
    return NULL;
}
//=====
//=====
const short CH_HIGHT_D=100, CH_BASE_D=470,
            CH_HIGHT=100, CH_BASE=450, t_pos=40;
//=====
double scale_fact=1;
//=====
void corr_win_proto(PCT win_source,PCT Proto, SCR_PNT win_size
                    ,short win_step,double CORR_THRESH,short *StripEnds)
{
    short i;
    char mess[40];
    short F=0;
    COLOR_VEC (*p_funct)(COLOR_RGB p1,double Thresh_mdl);
    p_funct=PointColFunct();

    PCT win(win_size.c,win_size.r);
    PCT tag(win_size.c,win_size.r);
    SCR_PNT st_t,st_win;
    AVERAGE_VEC middle_win[64],middle_tag;
    const AVERAGE_VEC z={{0,0,0},{0,0,0}};

    COLOR_VEC *corr_now,cr;
    const COLOR_VEC z_col={0.0,0.0,0.0};
    int line_size=win_source.s_cols+Proto.s_cols;
    //memory allocation

```

```

    if((corr_now= (COLOR_VEC*) malloc(
sizeof(COLOR_VEC)*(size_t)line_size*3))==NULL)
        {printf("WIN NOT MEMORY"); return;};

    st_t.r=0;
double dd;
    st_win.r=0;
    short k,FLG_COL=1;
    short StripStart,StripStop;
    short PartNum;
    k=PartNum=0;
    while(StripEnds[PartNum]>=0)
        {StripStart=StripEnds[PartNum++];
        StripStop=StripEnds[PartNum++];
        for (st_win.c=StripStart;
st_win.c+win_size.c<=StripStop;st_win.c+=win_step,k++)
            {
                FLG_COL=1;
                for(i=0;i<line_size;corr_now[i++]=z_col);

                win.load_template(win_source,st_win);
                middle_win[k]=average(win,Thresh_mdl,p_funct);

#ifdef MICI
#endif

const COLOR_VEC z_UNIT={1.0,1.0,1.0};

        for (st_t.c=0;st_t.c<=Proto.s_cols-win_size.c;st_t.c++)
            {
//=====
                tag.load_template(Proto,st_t);
                middle_tag=average(tag,Thresh_mdl,p_funct);
// DIFF ABS VALUES
#ifdef SINGL_VAL
                cr=template_conv_1( tag,win,Thresh_mdl,z_UNIT,p_funct);
                strcpy(mess," VECTOR Approach to CORRELATION ");
                corr_now[st_t.c]=Correlation_single_1(cr,middle_tag,middle_win[k],z_
UNIT);
#ifdef ABS_VALUE
                strcpy(mess," DIFF ABS VALUES/ max ABS VALUES");
                cr=
                    template_abs_diff_1 (tag,win,Thresh_mdl,z_UNIT,p_funct,
middle_tag,middle_win[k]);
#else
                cr=template_conv_1( tag,win,Thresh_mdl,z_UNIT,p_funct);
                strcpy(mess," PEARSON CORR. ");
                corr_now[st_t.c]=Correlation(cr,middle_tag,middle_win[k],z_UNIT);
#endif
            }
#endif

```

```

#endif

// ONLY LUMINANCE
//      strcat(mess," ALL 3 COMP");
//      strcat(mess," Only 0 COMP");
//      corr_now[st_t.c].c[1]=corr_now[st_t.c].c[2]=
//          corr_now[st_t.c].c[0];

#ifdef MICI
draw_color_corr_1(
corr_now[st_t.c],FLG_COL,CH_HIGHT_D,CH_BASE_D,CORR_THRESH,
st_t.c,Proto.s_cols);

FLG_COL=0;
#endif
}

//=====FILL PROTOCOL
//$ WILL BE USED AS SEPARATOR FOR READING
fprintf(PROTOCOL,"%t%s\t$t\t%4d\t $n",mess, st_win.c);
for(i=0;i<Proto.s_cols;i++) //ONLY 0 COMP
    fprintf(PROTOCOL,"%6g\t",corr_now[i].c[0]);
fprintf(PROTOCOL," \n");
}
}

win.free_PCT();
tag.free_PCT();
free((void *)corr_now);
return ;
}

//=====
=====

//=====
=====

void draw_chart(double *dist_line,short n,double max_value,
short CH_HIGHT,
short CH_BASE,double THRESH,
short t_pos)

{short i,j;
double p,
crit=max_value;
if(!max_value)
    for (i=0;i<n;i++)
        crit=(dist_line[i]>crit)? dist_line[i]:crit;
else crit=max_value;
if(!crit)
    crit=1;
p= CH_HIGHT*(1-THRESH/crit);
_moveto( 0,CH_BASE-CH_HIGHT );
}

```

```

SBRs = obj\MTCHSTR.sbr obj\VICALLOC.sbr obj\PROJCTN7.sbr
obj\L_TRN7.sbr\
    obj\PIC_M7.sbr obj\RES_MCH7.sbr obj\FILEMNP.sbr
obj\MTCHTPL2.sbr\
    obj\COR_FNC2.sbr

```

```
all: obj\$(PROJ).exe
```

```
.SUFFIXES:
```

```
.SUFFIXES:
```

```
.SUFFIXES: .obj .sbr .cpp
```

```

obj\MTCHSTR.obj : MTCHSTR.CPP C:\C700\INCLUDE\stdlib.h
C:\C700\INCLUDE\conio.h\
    C:\C700\INCLUDE\stdio.h C:\C700\INCLUDE\string.h\
    C:\C700\INCLUDE\graph.h C:\C700\INCLUDE\float.H
C:\C700\INCLUDE\time.h\
    C:\C700\INCLUDE\ctype.h C:\C700\INCLUDE\fstream.h\
    C:\C700\MFC\INCLUDE\afx.h matchng.h PIC_PRO.h MtchTpl2.h
shift.h\
    filemnp.h C:\C700\INCLUDE\iostream.h
C:\C700\MFC\INCLUDE\afx.in\
    c:\ilya\lib\mylib.h C:\C700\INCLUDE\direct.h ..\LIB\pic_mch7.h\
    ..\LIB\projctn7.h ..\LIB\res_mch7.h ..\LIB\lin_tm7.h\
    C:\C700\INCLUDE\ios.h C:\C700\INCLUDE\streamb.h\
    C:\C700\INCLUDE\istream.h C:\C700\INCLUDE\ostream.h\
    C:\C700\INCLUDE\vmemory.h C:\C700\INCLUDE\math.h
!IF $(DEBUG)
    @$ (CXX) @<<obj\$(PROJ).rsp
/c $(CXXFLAGS_G)
$(CXXFLAGS_D) /Foobj\MTCHSTR.obj MTCHSTR.CPP
<<
!ELSE
    @$ (CXX) @<<obj\$(PROJ).rsp
/c $(CXXFLAGS_G)
$(CXXFLAGS_R) /Foobj\MTCHSTR.obj MTCHSTR.CPP
<<
!ENDIF

```

```

obj\MTCHSTR.sbr : MTCHSTR.CPP C:\C700\INCLUDE\stdlib.h
C:\C700\INCLUDE\conio.h\
    C:\C700\INCLUDE\stdio.h C:\C700\INCLUDE\string.h\
    C:\C700\INCLUDE\graph.h C:\C700\INCLUDE\float.H
C:\C700\INCLUDE\time.h\
    C:\C700\INCLUDE\ctype.h C:\C700\INCLUDE\fstream.h\
    C:\C700\MFC\INCLUDE\afx.h matchng.h PIC_PRO.h MtchTpl2.h
shift.h\
    filemnp.h C:\C700\INCLUDE\iostream.h
C:\C700\MFC\INCLUDE\afx.in\

```

```

c:\ilya\lib\mylib.h C:\C700\INCLUDE\direct.h ..\LIB\pic_mch7.h\
..\LIB\projctn7.h ..\LIB\res_mch7.h ..\LIB\lin_tm7.h\
C:\C700\INCLUDE\ios.h C:\C700\INCLUDE\streamb.h\
C:\C700\INCLUDE\istream.h C:\C700\INCLUDE\ostream.h\
C:\C700\INCLUDE\vmemory.h C:\C700\INCLUDE\math.h

!IF $(DEBUG)
    @$(CXX) @<<obj\$(PROJ).rsp
/Zs $(CXXFLAGS_G)
$(CXXFLAGS_D) /Frobj\MTCHSTR.sbr MTCHSTR.CPP
<<
!ELSE
    @$(CXX) @<<obj\$(PROJ).rsp
/Zs $(CXXFLAGS_G)
$(CXXFLAGS_R) /Frobj\MTCHSTR.sbr MTCHSTR.CPP
<<
!ENDIF

obj\VICALLOC.obj : ..\LIB\VICALLOC.CPP C:\C700\INCLUDE\stdlib.h\
C:\C700\INCLUDE\stdio.h C:\C700\INCLUDE\vmemory.h\
C:\C700\INCLUDE\malloc.h
!IF $(DEBUG)
    @$(CXX) @<<obj\$(PROJ).rsp
/c $(CXXFLAGS_G)
$(CXXFLAGS_D) /Fobj\VICALLOC.obj ..\LIB\VICALLOC.CPP
<<
!ELSE
    @$(CXX) @<<obj\$(PROJ).rsp
/c $(CXXFLAGS_G)
$(CXXFLAGS_R) /Fobj\VICALLOC.obj ..\LIB\VICALLOC.CPP
<<
!ENDIF

obj\VICALLOC.sbr : ..\LIB\VICALLOC.CPP C:\C700\INCLUDE\stdlib.h\
C:\C700\INCLUDE\stdio.h C:\C700\INCLUDE\vmemory.h\
C:\C700\INCLUDE\malloc.h
!IF $(DEBUG)
    @$(CXX) @<<obj\$(PROJ).rsp
/Zs $(CXXFLAGS_G)
$(CXXFLAGS_D) /Frobj\VICALLOC.sbr ..\LIB\VICALLOC.CPP
<<
!ELSE
    @$(CXX) @<<obj\$(PROJ).rsp
/Zs $(CXXFLAGS_G)
$(CXXFLAGS_R) /Frobj\VICALLOC.sbr ..\LIB\VICALLOC.CPP
<<
!ENDIF

obj\PROJECTN7.obj : ..\LIB\PROJECTN7.CPP C:\C700\INCLUDE\graph.h\

```

```

SBRS = obj\MTCHSTR.sbr obj\VICALLOC.sbr obj\PROJECTN7.sbr
obj\L_TRN7.sbr\
      obj\PIC_M7.sbr obj\RES_MCH7.sbr obj\FILEMNP.sbr
obj\MTCHTPL2.sbr\
      obj\COR_FNC2.sbr

```

```
all: obj\$(PROJ).exe
```

```

.SUFFIXES:
.SUFFIXES:
.SUFFIXES: .obj .sbr .cpp

```

```

obj\MTCHSTR.obj : MTCHSTR.CPP C:\C700\INCLUDE\stdlib.h
C:\C700\INCLUDE\conio.h\
      C:\C700\INCLUDE\stdio.h C:\C700\INCLUDE\string.h\
      C:\C700\INCLUDE\graph.h C:\C700\INCLUDE\float.h
C:\C700\INCLUDE\time.h\
      C:\C700\INCLUDE\ctype.h C:\C700\INCLUDE\fstream.h\
      C:\C700\MFC\INCLUDE\afx.h matchng.h PIC_PRO.h MchTpl2.h
shift.h\

```

```

      filemnp.h C:\C700\INCLUDE\iostream.h
C:\C700\MFC\INCLUDE\afx.in\
      c:\ilya\lib\mylib.h C:\C700\INCLUDE\direct.h ..\LIB\pic_mch7.h\
      ..\LIB\projctn7.h ..\LIB\res_mch7.h ..\LIB\lin_tm7.h\
      C:\C700\INCLUDE\ios.h C:\C700\INCLUDE\streamb.h\
      C:\C700\INCLUDE\istream.h C:\C700\INCLUDE\ostream.h\
      C:\C700\INCLUDE\vmemory.h C:\C700\INCLUDE\math.h

```

```

!IF $(DEBUG)
    @$ (CXX) @<<obj\$(PROJ).rsp
/c $(CXXFLAGS_G)
$(CXXFLAGS_D) /Foobj\MTCHSTR.obj MTCHSTR.CPP
<<
!ELSE
    @$ (CXX) @<<obj\$(PROJ).rsp
/c $(CXXFLAGS_G)
$(CXXFLAGS_R) /Foobj\MTCHSTR.obj MTCHSTR.CPP
<<
!ENDIF

```

```

obj\MTCHSTR.sbr : MTCHSTR.CPP C:\C700\INCLUDE\stdlib.h
C:\C700\INCLUDE\conio.h\
      C:\C700\INCLUDE\stdio.h C:\C700\INCLUDE\string.h\
      C:\C700\INCLUDE\graph.h C:\C700\INCLUDE\float.h
C:\C700\INCLUDE\time.h\
      C:\C700\INCLUDE\ctype.h C:\C700\INCLUDE\fstream.h\
      C:\C700\MFC\INCLUDE\afx.h matchng.h PIC_PRO.h MchTpl2.h
shift.h\
      filemnp.h C:\C700\INCLUDE\iostream.h
C:\C700\MFC\INCLUDE\afx.in\

```

```

c:\iyallib\mylib.h C:\C700\INCLUDE\direct.h ..\LIB\pic_mch7.h\
..\LIB\projctn7.h ..\LIB\res_mch7.h ..\LIB\lin_tm7.h\
C:\C700\INCLUDE\ios.h C:\C700\INCLUDE\streamb.h\
C:\C700\INCLUDE\istream.h C:\C700\INCLUDE\ostream.h\
C:\C700\INCLUDE\vmemory.h C:\C700\INCLUDE\math.h

!IF $(DEBUG)
    @$(CXX) @<<obj\$(PROJ).rsp
/Zs $(CXXFLAGS_G)
$(CXXFLAGS_D) /FRobj\MTCHSTR.sbr MTCHSTR.CPP
<<
!ELSE
    @$(CXX) @<<obj\$(PROJ).rsp
/Zs $(CXXFLAGS_G)
$(CXXFLAGS_R) /FRobj\MTCHSTR.sbr MTCHSTR.CPP
<<
!ENDIF

obj\VICALLOC.obj : ..\LIB\VICALLOC.CPP C:\C700\INCLUDE\stdlib.h\
C:\C700\INCLUDE\stdio.h C:\C700\INCLUDE\vmemory.h\
C:\C700\INCLUDE\malloc.h
!IF $(DEBUG)
    @$(CXX) @<<obj\$(PROJ).rsp
/c $(CXXFLAGS_G)
$(CXXFLAGS_D) /Foobj\VICALLOC.obj ..\LIB\VICALLOC.CPP
<<
!ELSE
    @$(CXX) @<<obj\$(PROJ).rsp
/c $(CXXFLAGS_G)
$(CXXFLAGS_R) /Foobj\VICALLOC.obj ..\LIB\VICALLOC.CPP
<<
!ENDIF

obj\VICALLOC.sbr : ..\LIB\VICALLOC.CPP C:\C700\INCLUDE\stdlib.h\
C:\C700\INCLUDE\stdio.h C:\C700\INCLUDE\vmemory.h\
C:\C700\INCLUDE\malloc.h
!IF $(DEBUG)
    @$(CXX) @<<obj\$(PROJ).rsp
/Zs $(CXXFLAGS_G)
$(CXXFLAGS_D) /FRobj\VICALLOC.sbr ..\LIB\VICALLOC.CPP
<<
!ELSE
    @$(CXX) @<<obj\$(PROJ).rsp
/Zs $(CXXFLAGS_G)
$(CXXFLAGS_R) /FRobj\VICALLOC.sbr ..\LIB\VICALLOC.CPP
<<
!ENDIF

obj\PROJECTN7.obj : ..\LIB\PROJECTN7.CPP C:\C700\INCLUDE\graph.h\

```



```

C:\C700\INCLUDE\stdlib.h C:\C700\INCLUDE\iostream.h
..\LIB\projctn7.h\
C:\C700\INCLUDE\ios.h C:\C700\INCLUDE\streamb.h\
C:\C700\INCLUDE\istream.h C:\C700\INCLUDE\ostream.h\
C:\C700\INCLUDE\math.h
!IF $(DEBUG)
    @$(CXX) @<<obj\$(PROJ).rsp
/c $(CXXFLAGS_G)
$(CXXFLAGS_D) /Foobj\PROJCTN7.obj ..\LIB\PROJCTN7.CPP
<<
!ELSE
    @$(CXX) @<<obj\$(PROJ).rsp
/c $(CXXFLAGS_G)
$(CXXFLAGS_R) /Foobj\PROJCTN7.obj ..\LIB\PROJCTN7.CPP
<<
!ENDIF

obj\PROJCTN7.sbr : ..\LIB\PROJCTN7.CPP C:\C700\INCLUDE\graph.h\
C:\C700\INCLUDE\stdlib.h C:\C700\INCLUDE\iostream.h
..\LIB\projctn7.h\
C:\C700\INCLUDE\ios.h C:\C700\INCLUDE\streamb.h\
C:\C700\INCLUDE\istream.h C:\C700\INCLUDE\ostream.h\
C:\C700\INCLUDE\math.h
!IF $(DEBUG)
    @$(CXX) @<<obj\$(PROJ).rsp
/Zs $(CXXFLAGS_G)
$(CXXFLAGS_D) /FRobj\PROJCTN7.sbr ..\LIB\PROJCTN7.CPP
<<
!ELSE
    @$(CXX) @<<obj\$(PROJ).rsp
/Zs $(CXXFLAGS_G)
$(CXXFLAGS_R) /FRobj\PROJCTN7.sbr ..\LIB\PROJCTN7.CPP
<<
!ENDIF

obj\L_TRN7.obj : ..\LIB\L_TRN7.CPP C:\C700\INCLUDE\stdlib.h\
C:\C700\INCLUDE\stdio.h C:\C700\INCLUDE\conio.h\
C:\C700\INCLUDE\malloc.h ..\LIB\vicalloc.h ..\LIB\lin_trn7.h\
C:\C700\INCLUDE\memory.h ..\LIB\projctn7.h ..\LIB\res_mch7.h\
..\LIB\pic_mch7.h C:\C700\INCLUDE\math.h
C:\C700\INCLUDE\graph.h
!IF $(DEBUG)
    @$(CXX) @<<obj\$(PROJ).rsp
/c $(CXXFLAGS_G)
$(CXXFLAGS_D) /Foobj\L_TRN7.obj ..\LIB\L_TRN7.CPP
<<
!ELSE
    @$(CXX) @<<obj\$(PROJ).rsp
/c $(CXXFLAGS_G)

```

```

$(CXXFLAGS_R) /Foobj\L_TRN7.obj ..\LIB\L_TRN7.CPP
<<
!ENDIF

obj\L_TRN7.sbr : ..\LIB\L_TRN7.CPP C:\C700\INCLUDE\stdlib.h\
C:\C700\INCLUDE\stdio.h C:\C700\INCLUDE\conio.h\
C:\C700\INCLUDE\malloc.h ..\LIB\vicalloc.h ..\LIB\lin_trn7.h\
C:\C700\INCLUDE\vmemory.h ..\LIB\projctn7.h ..\LIB\res_mch7.h\
..\LIB\pic_mch7.h C:\C700\INCLUDE\math.h
C:\C700\INCLUDE\graph.h
!IF $(DEBUG)
    @$(CXX) @<<obj\$(PROJ).rsp
/Zs $(CXXFLAGS_G)
$(CXXFLAGS_D) /Fobj\L_TRN7.sbr ..\LIB\L_TRN7.CPP
<<
!ELSE
    @$(CXX) @<<obj\$(PROJ).rsp
/Zs $(CXXFLAGS_G)
$(CXXFLAGS_R) /Fobj\L_TRN7.sbr ..\LIB\L_TRN7.CPP
<<
!ENDIF

obj\PIC_M7.obj : ..\LIB\PIC_M7.CPP C:\C700\INCLUDE\stdlib.h\
C:\C700\INCLUDE\stdio.h C:\C700\INCLUDE\graph.h
C:\C700\INCLUDE\math.h\
C:\C700\INCLUDE\io.h C:\C700\INCLUDE\fcntl.h
C:\C700\INCLUDE\string.h\
C:\C700\INCLUDE\float.H C:\C700\INCLUDE\malloc.h ..\LIB\phdr.h\
..\LIB\vicalloc.h ..\LIB\pic_mch7.h C:\C700\INCLUDE\vmemory.h\
..\LIB\projctn7.h
!IF $(DEBUG)
    @$(CXX) @<<obj\$(PROJ).rsp
/c $(CXXFLAGS_G)
$(CXXFLAGS_D) /Foobj\PIC_M7.obj ..\LIB\PIC_M7.CPP
<<
!ELSE
    @$(CXX) @<<obj\$(PROJ).rsp
/c $(CXXFLAGS_G)
$(CXXFLAGS_R) /Foobj\PIC_M7.obj ..\LIB\PIC_M7.CPP
<<
!ENDIF

obj\PIC_M7.sbr : ..\LIB\PIC_M7.CPP C:\C700\INCLUDE\stdlib.h\
C:\C700\INCLUDE\stdio.h C:\C700\INCLUDE\graph.h
C:\C700\INCLUDE\math.h\
C:\C700\INCLUDE\io.h C:\C700\INCLUDE\fcntl.h
C:\C700\INCLUDE\string.h\
C:\C700\INCLUDE\float.H C:\C700\INCLUDE\malloc.h ..\LIB\phdr.h\
..\LIB\vicalloc.h ..\LIB\pic_mch7.h C:\C700\INCLUDE\vmemory.h\

```

```

        ..\LIB\projctn7.h
!!IF $(DEBUG)
    @$(CXX) @<<obj\$(PROJ).rsp
/Zs $(CXXFLAGS_G)
$(CXXFLAGS_D) /FRobj\PIC_M7.sbr ..\LIB\PIC_M7.CPP
<<
!ELSE
    @$(CXX) @<<obj\$(PROJ).rsp
/Zs $(CXXFLAGS_G)
$(CXXFLAGS_R) /FRobj\PIC_M7.sbr ..\LIB\PIC_M7.CPP
<<
!ENDIF

obj\RES_MCH7.obj : ..\LIB\RES_MCH7.CPP C:\C700\INCLUDE\stdlib.h\
    C:\C700\INCLUDE\vmemory.h ..\LIB\pic_mch7.h ..\LIB\res_mch7.h\
    C:\C700\INCLUDE\graph.h ..\LIB\projctn7.h C:\C700\INCLUDE\math.h
!!IF $(DEBUG)
    @$(CXX) @<<obj\$(PROJ).rsp
/c $(CXXFLAGS_G)
$(CXXFLAGS_D) /Foobj\RES_MCH7.obj ..\LIB\RES_MCH7.CPP
<<
!ELSE
    @$(CXX) @<<obj\$(PROJ).rsp
/c $(CXXFLAGS_G)
$(CXXFLAGS_R) /Foobj\RES_MCH7.obj ..\LIB\RES_MCH7.CPP
<<
!ENDIF

obj\RES_MCH7.sbr : ..\LIB\RES_MCH7.CPP C:\C700\INCLUDE\stdlib.h\
    C:\C700\INCLUDE\vmemory.h ..\LIB\pic_mch7.h ..\LIB\res_mch7.h\
    C:\C700\INCLUDE\graph.h ..\LIB\projctn7.h C:\C700\INCLUDE\math.h
!!IF $(DEBUG)
    @$(CXX) @<<obj\$(PROJ).rsp
/Zs $(CXXFLAGS_G)
$(CXXFLAGS_D) /FRobj\RES_MCH7.sbr ..\LIB\RES_MCH7.CPP
<<
!ELSE
    @$(CXX) @<<obj\$(PROJ).rsp
/Zs $(CXXFLAGS_G)
$(CXXFLAGS_R) /FRobj\RES_MCH7.sbr ..\LIB\RES_MCH7.CPP
<<
!ENDIF

obj\FILEMNP.obj : FILEMNP.CPP C:\C700\INCLUDE\stdlib.h\
    C:\C700\INCLUDE\string.h C:\C700\INCLUDE\io.h\
    C:\C700\MFC\INCLUDE\afx.h C:\C700\INCLUDE\direct.h\
    C:\C700\INCLUDE\stdio.h C:\C700\INCLUDE\time.h\
    C:\C700\MFC\INCLUDE\afx.inl
!!IF $(DEBUG)

```

```

        @$(CXX) @<<obj\$(PROJ).rsp
/c $(CXXFLAGS_G)
$(CXXFLAGS_D) /Foobj\FILEMNP.obj FILEMNP.CPP
<<
ELSE
        @$(CXX) @<<obj\$(PROJ).rsp
/c $(CXXFLAGS_G)
$(CXXFLAGS_R) /Foobj\FILEMNP.obj FILEMNP.CPP
<<
ENDIF

obj\FILEMNP.sbr : FILEMNP.CPP C:\C700\INCLUDE\stdlib.h\
C:\C700\INCLUDE\string.h C:\C700\INCLUDE\io.h\
C:\C700\MFC\INCLUDE\afx.h C:\C700\INCLUDE\direct.h\
C:\C700\INCLUDE\stdio.h C:\C700\INCLUDE\time.h\
C:\C700\MFC\INCLUDE\afx.inl
!IF $(DEBUG)
        @$(CXX) @<<obj\$(PROJ).rsp
/Zs $(CXXFLAGS_G)
$(CXXFLAGS_D) /FRobj\FILEMNP.sbr FILEMNP.CPP
<<
ELSE
        @$(CXX) @<<obj\$(PROJ).rsp
/Zs $(CXXFLAGS_G)
$(CXXFLAGS_R) /FRobj\FILEMNP.sbr FILEMNP.CPP
<<
ENDIF

obj\MTCHTPL2.obj : MTCHTPL2.CPP C:\C700\INCLUDE\memory.h\
C:\C700\INCLUDE\stdio.h C:\C700\INCLUDE\stdlib.h\
C:\C700\INCLUDE\conio.h C:\C700\INCLUDE\malloc.h\
C:\C700\INCLUDE\float.h C:\C700\INCLUDE\graph.h\
C:\C700\INCLUDE\string.h matchng.h MtchTpl2.h cor_fnc2.h\
c:\ilya\lib\mylib.h ..\LIB\projctn7.h ..\LIB\pic_mch7.h\
..\LIB\res_mch7.h PIC_PRO.h ..\LIB\in_tm7.h
C:\C700\INCLUDE\math.h\
C:\C700\INCLUDE\direct.h C:\C700\MFC\INCLUDE\afx.h filemnp.h\
C:\C700\INCLUDE\time.h C:\C700\MFC\INCLUDE\afx.inl
!IF $(DEBUG)
        @$(CXX) @<<obj\$(PROJ).rsp
/c $(CXXFLAGS_G)
$(CXXFLAGS_D) /Foobj\MTCHTPL2.obj MTCHTPL2.CPP
<<
ELSE
        @$(CXX) @<<obj\$(PROJ).rsp
/c $(CXXFLAGS_G)
$(CXXFLAGS_R) /Foobj\MTCHTPL2.obj MTCHTPL2.CPP
<<
ENDIF

```

```

obj\MTCHTPL2.sbr : MTCHTPL2.CPP C:\C700\INCLUDE\vmemory.h\
C:\C700\INCLUDE\stdio.h C:\C700\INCLUDE\stdlib.h\
C:\C700\INCLUDE\conio.h C:\C700\INCLUDE\malloc.h\
C:\C700\INCLUDE\float.h C:\C700\INCLUDE\graph.h\
C:\C700\INCLUDE\string.h matchng.h MtchTpl2.h cor_fnc2.h\
c:\ilya\lib\mylib.h ..\LIB\projctn7.h ..\LIB\pic_mch7.h\
..\LIB\res_mch7.h PIC_PRO.h ..\LIB\in_tm7.h
C:\C700\INCLUDE\math.h\
C:\C700\INCLUDE\direct.h C:\C700\MFC\INCLUDE\afx.h filemnp.h\
C:\C700\INCLUDE\time.h C:\C700\MFC\INCLUDE\afx.inl
!IF $(DEBUG)
    @$(CXX) @<<obj\$(PROJ).rsp
/Zs $(CXXFLAGS_G)
$(CXXFLAGS_D) /FRobj\MTCHTPL2.sbr MTCHTPL2.CPP
<<
!ELSE
    @$(CXX) @<<obj\$(PROJ).rsp
/Zs $(CXXFLAGS_G)
$(CXXFLAGS_R) /FRobj\MTCHTPL2.sbr MTCHTPL2.CPP
<<
!ENDIF

obj\COR_FNC2.obj : COR_FNC2.CPP C:\C700\INCLUDE\stdio.h\
C:\C700\INCLUDE\stdlib.h C:\C700\INCLUDE\conio.h\
C:\C700\INCLUDE\float.h C:\C700\INCLUDE\graph.h cor_fnc2.h\
..\LIB\pic_mch7.h ..\LIB\res_mch7.h C:\C700\INCLUDE\vmemory.h\
..\LIB\projctn7.h C:\C700\INCLUDE\math.h
!IF $(DEBUG)
    @$(CXX) @<<obj\$(PROJ).rsp
/c $(CXXFLAGS_G)
$(CXXFLAGS_D) /Foobj\COR_FNC2.obj COR_FNC2.CPP
<<
!ELSE
    @$(CXX) @<<obj\$(PROJ).rsp
/c $(CXXFLAGS_G)
$(CXXFLAGS_R) /Foobj\COR_FNC2.obj COR_FNC2.CPP
<<
!ENDIF

obj\COR_FNC2.sbr : COR_FNC2.CPP C:\C700\INCLUDE\stdio.h\
C:\C700\INCLUDE\stdlib.h C:\C700\INCLUDE\conio.h\
C:\C700\INCLUDE\float.h C:\C700\INCLUDE\graph.h cor_fnc2.h\
..\LIB\pic_mch7.h ..\LIB\res_mch7.h C:\C700\INCLUDE\vmemory.h\
..\LIB\projctn7.h C:\C700\INCLUDE\math.h
!IF $(DEBUG)
    @$(CXX) @<<obj\$(PROJ).rsp
/Zs $(CXXFLAGS_G)
$(CXXFLAGS_D) /FRobj\COR_FNC2.sbr COR_FNC2.CPP

```

```

<<
!ELSE
    @$(CXX) @<<obj$(PROJ).rsp
/Zs $(CXXFLAGS_G)
$(CXXFLAGS_R) /Frobj\COR_FNC2.sbr COR_FNC2.CPP
<<
!ENDIF

obj$(PROJ).bsc : $(SBRs)
    $(BSCMAKE) @<<
$(BRFLAGS) $(SBRs)
<<

obj$(PROJ).exe : $(OBJS)
    -$(NMAKEBSC1) MAKEFLAGS=
    -$(NMAKEBSC2) $(NMFLAGS) -f $(PROJFILE) obj$(PROJ).bsc
!IF $(DEBUG)
    $(LRF) @<<obj$(PROJ).lrf
$(RT_OBJS: = +^
) $(OBJS: = +^
)
$@
$(MAPFILE_D)
$(LIBS: = +^
) +
$(LLIBS_G: = +^
) +
$(LLIBS_D: = +^
)
$(DEF_FILE) $(LFLAGS_G) $(LFLAGS_D);
<<
!ELSE
    $(LRF) @<<obj$(PROJ).lrf
$(RT_OBJS: = +^
) $(OBJS: = +^
)
$@
$(MAPFILE_R)
$(LIBS: = +^
) +
$(LLIBS_G: = +^
) +
$(LLIBS_R: = +^
)
$(DEF_FILE) $(LFLAGS_G) $(LFLAGS_R);
<<
!ENDIF
$(LINKER) @obj$(PROJ).lrf

```

```
.cpp.obj :
!!IF $(DEBUG)
    @$(CXX) @<<obj\$(PROJ).rsp
/c $(CXXFLAGS_G)
$(CXXFLAGS_D) /Fo$@ $<
<<
!ELSE
    @$(CXX) @<<obj\$(PROJ).rsp
/c $(CXXFLAGS_G)
$(CXXFLAGS_R) /Fo$@ $<
<<
!ENDIF
```

```
.cpp.sbr :
!!IF $(DEBUG)
    @$(CXX) @<<obj\$(PROJ).rsp
/Zs $(CXXFLAGS_G)
$(CXXFLAGS_D) /FR$@ $<
<<
!ELSE
    @$(CXX) @<<obj\$(PROJ).rsp
/Zs $(CXXFLAGS_G)
$(CXXFLAGS_R) /FR$@ $<
<<
!ENDIF
```

```
run: obj\$(PROJ).exe
    obj\$(PROJ).exe $(RUNFLAGS)
```

```
debug: obj\$(PROJ).exe
    CV $(CVFLAGS) obj\$(PROJ).exe $(RUNFLAGS)
```

```
//return file name without extwntion in "name" and TRUE 1 if file exist;
int next_pict(char *name,char *mask,char *ext,int num);
//if num=-2 initialisation;
//    -1 next
//    >0 adding this # to mask
//    and reinitialise to this #
// NULL if notexist file with ".ext"

//=====
short MaskDecoding(char *msk);
//=====
//=====
CString MakeName(char *p );
CString MakeName(CString N);
//=====
```



```
#ifndef MATCHNG
#define MATCHNG
#include "mylib.h"

#define EXTANTION ".sg2"

#define MaxProtoNum 40
#define MaxWinNum 40
#define PRESENT_HIGHT 32
#define MaxSignSize 256
//
#define CALCULATION_HIGHT 16

#endif
```

```

#ifndef TEMPLT
#define MchTPLT
#include "projctn7.h"
#include "pic_mch7.h"
#include "res_mch7.h"
#include "pic_pro.h"
#define NTSC 0
#define HSI 1
#define New_plan 2
#define RGB 3
#define LUMIN_THR 4
#define IHS 5
class RsltNow
{public:
    short voices; // voiting numbers
    double value; //value
    RsltNow::RsltNow(){voices=0;value=0;}
    RsltNow::RsltNow(short d,double v){voices=d;value=v;}
};
//=====
class RSLT
{ public:
    RsltNow R;
    short pos; // position in string
    short ShNumb;
    short ProtoNum;
    short StrNum;
    RSLT::RSLT(){};
    RSLT::RSLT(short num,short p, double v,
        short shft,short pro,short st)
    {R=RsltNow::RsltNow(num,v);
    pos=p;
    ShNumb=shft;
    ProtoNum=pro;
    StrNum=st;
    }
};
//=====
double CalcCorrThresh(short * Hst,short HDim, short NPntAboveThr,
    int PlusThresh, double PrcntLvl=0);
double CorrelationEstim(double C, double MinVal,double MaxVal,
void*AddInf=NULL);

void MatchForProtoStr(PCT &T,PRT &P,SCR_PNT winsize,short winstep,
    double *CorrThresh,RsltNow *NowRslt,
    short *StripEnds);

#endif

```

```

#ifndef PIC_PRO
#define PIC_PRO
#include <stdlib.h>
#include <direct.h>
#include <afx.h>
#include <pic_mch7.h>
#include "filemnp.h"
#define STR_MAX 4
//=====
=====
const SCR_PNT z_0(0,0);
class PRT:public PCT
{public:
//information
    CString PathName;
    CString FRAME_Number;
    CString STRING_name;
    CString SIGN_name;
    short Pos; // Position in the string
    long NumberOfChk,MaxNum;
    double *Charact;
//models
    PRT::~PRT()
        {this->free_PCT();
        Pos=0;
        if(MaxNum)
            delete Charact;
        Charact=NULL;
        MaxNum=NumberOfChk=0;
        }
//-----
    PRT::PRT()
        {NumberOfChk=MaxNum=s_cols=s_rows=0;
        Charact=NULL;pict=NULL;
        }
//-----
    PRT::PRT (short n_cols, short n_rows)
        {*(PCT *)this=PCT::PCT(n_cols,n_rows);
        NumberOfChk=MaxNum=0;
        Charact=NULL;
        }
//=====
int read_proto_SGN(char ext[]=".sgn")
{
    CString new_name(' ',80);

    PathName=MakeName(PathName);
    new_name=PathName+ext;
    char now[80];

```

```

FILE *datfp;
if(!(datfp=fopen((const char*)new_name,"r"))) return 1;
if(fscanf(datfp,"%*[^\\n]s")==EOF)goto ERR;
if(fscanf(datfp,"%s ",now)==EOF)goto ERR;
if(fscanf(datfp,"%s ",now)==EOF)goto ERR;
if(fscanf(datfp,"%s ",now)==EOF)goto
ERR;FRAME_Number=now;
if(fscanf(datfp,"%s ",now)==EOF)goto
ERR;STRING_name=now;
if(fscanf(datfp,"%s ",now)==EOF)goto ERR; SIGN_name=now;
FRAME_Number.MakeUpper();
STRING_name.MakeUpper();
SIGN_name.MakeUpper();
fclose(datfp);
return 0;
ERR:fclose (datfp); return 1;
}
//=====
int proto_storage_rgb(char *name,struct _videoconfig vc,char *ext=".sgn")
{*(PCT *)this=sign_storage_rgb(name,vc);
if (!s_cols) return 1;
PathName=MakeName(name);
if (read_proto_SGN(ext))
{free_PCT();
return 1;
}
return 0;
}
//-----
int read_proto_DBC(FILE *datfp)
{
char now[80];
if(fscanf(datfp,"%s ",now)==EOF)goto
ERR;PathName=MakeName(now);
if(fscanf(datfp,"%s ",now)==EOF)goto
ERR;FRAME_Number=now;
if(fscanf(datfp,"%s ",now)==EOF)goto
ERR;STRING_name=now;
if(fscanf(datfp,"%s ",now)==EOF)goto ERR; SIGN_name=now;
if(fscanf(datfp,"%d ",&(this->s_cols))==EOF)goto ERR;
FRAME_Number.MakeUpper();
STRING_name.MakeUpper();
SIGN_name.MakeUpper();
return 1;
ERR: return 0;
}
//=====
===
int alloc_Charact_dbl(long Num)

```

```

{
    if(!(Charact=new double[Num])) return 1;
    MaxNum=Num; NumberOfChk=0;
    return 0;
}
//-----
void free_Charact()
{delete Charact;
  Charact=NULL;
}
//-----
int read_Charact_dbl(FILE *inp,long Num)
{short i;
  double d;
  if(MaxNum<(NumberOfChk+Num)) return 1;
  for (i=0;i<Num;i++)
  {if(fscanf(inp,"%lf",&d) ==EOF) return 1;
    if(fabs(d)<1.0e-4) d=0;
    Charact[NumberOfChk]=d;
    NumberOfChk++;
  }
  return 0;
}
//-----
double CorrValue(short WNum,short Pnum)
{return (*(Charact+(long)WNum*s_cols+Pnum));
}
//=====
=====
//=====RETURN NUMBER OF STRIPS
int read_target_SGN(SCR_PNT vrt[][4],char ext[]=".sgs")
{int n=0,j,FLG,s;
  CString new_name(' ',80);

  PathName=MakeName(PathName);
  new_name=PathName+ext;
  char now[80];
  FILE *datfp;
  if(!(datfp=fopen((const char*)new_name,"r"))) return 1;
  if(fscanf(datfp,"%*[^\n]s")==EOF)goto OUT;
  if(fscanf(datfp,"%s ",now)==EOF)goto OUT;
  if(fscanf(datfp,"%s ",now)==EOF)goto OUT;
  if(fscanf(datfp,"%s ",now)==EOF)goto
OUT;STRING_name=now;
  if(fscanf(datfp,"%s ",now)==EOF)goto OUT; SIGN_name=now;
  if((s=PathName.ReverseFind("\\"))<0)
    s=PathName.ReverseFind('.');
  FRAME_Number=
    PathName.Right(PathName.GetLength()-s);
}

```

```

        STRING_name.MakeUpper();
        SIGN_name.MakeUpper();
    do{
        for(j=0;j<4;j++)
            if((FLG=fscanf(datfp,"%d %d ",&(vrt[n][j].c),&(vrt[n][j].r)))==EOF)
                goto OUT;
        n++;
    }
    while(n<STR_MAX-1);
    OUT:fclose (datfp); return n;
}
//-----
};
//=====

#define Unknown -1
//=====
=
typedef struct
{
    short n; // voiting numbers
    short pos; // position in string
    double value; //value
} RSLT_old;

//-----
void HistCollect(short NOfWin,short St,short Fin,PRT &Db);
RSLT_old LineEstimation (short TagSize, PRT &Db,short NOfWin,
                        short WSize,double Thr);
int LineInP(const PRT &P, PRT T, short rw, short Xpos,struct _videoconfig vc);
double LinInter( PRT &P,short WNum,short WSize ,double Pt);
void HistThresh(short *H,short *BotThr,short *TopThr,short num);

#endif

```

```

#include "lin_tm7.h"
double VShift[4][4]= {
    {0.0, 0.0, 0.0, 0.0}
    ,{-2,+2,+2,-2}
    ,{+2,-2,-2,+2}
    // ,{0.125,0,0,0.125}
    // ,{-0.125,0,0,-0.125}
    // ,{0,0.125,0.125,0}
    // ,{0,-0.125,-0.125,0}
    // ,{0.125,0.125,0.125,0.125}
//      ,{-0.125,-0.125,-0.125,-0.125}
//      };

//=====      ,{0.1875,0,0,0.1875}
//=====      ,{-0.1875,0,0,-0.1875}
//=====      {0,0.1875,0.1875,0}
//=====      {0,-0.1875,-0.1875,0}
//      ,{0.1875,0.1875,0.1875,0.1875}
//      ,{-0.1875,-0.1875,-0.1875,-0.1875}
//      ,{0.35,0.35,0.35,0.35}
//      };

void VrtxCalculation(PRT &T,double sh[4],SCR_PNT NewV[4],
                    SCR_PNT OldV[4])
{short j,k,MaxV;
  MaxV=T.s_rows;
  match_vertex( OldV);
  SCR_PNT v[4];
/*=====
  DIR_LINE top(OldV[3],OldV[0]),down(OldV[1],OldV[2]);
  DIR_LINE top_new,down_new;
  vert_for_map(*((PCT *)(&T)),top,down,&top_new,&down_new);
  PT st1 =down_new.Start_p(),st2=top_new.Start_p(),
  end1 =down_new.End_p(),end2=top_new.End_p();

  PT_SCR(st2,v[3]);
  PT_SCR(end2,v[0]);
  PT_SCR(st1,v[1]);
  PT_SCR(end1,v[2]);
  match_vertex( v);
  =====*/
  v[0]=OldV[0];v[1]=OldV[1];v[2]=OldV[2];v[3]=OldV[3];
  //=====
  for(j=0;j<4;j++)
  {
    NewV[j].c=v[j].c;
    k=((j%2)?j-1:j+1);
    if (fabs(sh[j])<1)
      {NewV[j].r=(sh[j]?(short)((v[j].r-v[k].r)*sh[j])+

```

```

        v[j].r=v[j].r);
    }
    else
        NewV[j].r=v[j].r+sh[j];
    NewV[j].r=__max(0, __min(NewV[j].r,MaxV-1));
}
while((NewV[1].r-NewV[0].r)<8)
{NewV[1].r=__min(NewV[1].r++,MaxV-1);
  if((NewV[1].r-NewV[0].r)<8)
    NewV[0].r=__max(0, NewV[0].r-);
}
while((NewV[2].r-NewV[3].r)<8)
{NewV[2].r=__min(NewV[2].r++,MaxV-1);
  if((NewV[2].r-NewV[3].r)<8)
    NewV[3].r=__max(0, NewV[3].r-);
}
}
//=====
=====

```



```
#include <stdlib.h>
#include <string.h>
#include <io.h>
#include <afx.h>
#include <direct.h>
```

```
//return file name without extwntion in "name" and TRUE 1 if file exist;
int next_pict(char *name,char *mask,char *ext,int num)
```

```
//if num=-2 initalisation;
//      -1 next
//      >0 adding this # to mask
//      and reinialise to this #
// NULL if notexist file with ".ext"
```

```
{static int now;
 char full_name[80];
 strcpy(name,mask);
 if (num==2)now=0;
 else if (num==1)
     now++;
 else if (num<0) return 0;
 else now=num;
 _itoa( now, name+strlen(name), 10);
 strcat(strcpy(full_name,name),ext);
 //1 if file exist
 return(!_access( full_name, 0 ));
}
```

```
//=====
=====
```

```
short MaskDecoding(char *msk)
{char *p,nm[40];
 strcpy(nm,msk);
 if(p=strchr( nm, (int) '\\ ' ))
     p+=2;
 else
     if(p=strchr( nm, (int) ':' ))
         p+=2;
 else
     p=nm+1;
 *(p+2)='\0';
 return ((short)atoi( p ));
}
```

```
//=====
```

```
CString MakeName(CString N)
{
 short k=(N.SpanIncluding("\\t")).GetLength();
 char *p,fp[80];
 p=((char*)(const char *)N)+k;
 CString M=p;
```

```
if(M.Find('.')<0)
{if(M.GetAt(0)=='\\')
{ M+='.'+M;
  M= (char)(_getdrive( )-1+'A')+M;
}
else
  M=_fullpath(fp,(const char *)M,80);
}
```

```
M.MakeLower();
```

```
return M;
```

```
}
```

```
//=====
```

```
=
```

```
CString MakeName(char *p )
```

```
{CString M(p);
```

```
return (MakeName(M));
```

```
}
```

```
//=====
```

```
=====
```

```

#include <vmemory.h>
#include <stdio.h>
#include <stdlib.h>
#include <conio.h>
#include <malloc.h>
#include <float.h>
#include <graph.h>
#include <string.h>

#include "matchng.h"
#include "mtchtpl2.h"
#include "cor_fnc2.h"

#define MAX_LINE 1024

//extern short NumberOfWin;
extern double GAMMA,Thresh_mdl;
extern short MAP;
extern short VOITING;
extern struct _videoconfig vc;

#define ProbValue(S,x)  (2*(double)(x)/((S)-1.0)-1.0)
void draw_color_corr_1(COLOR_VEC corr,short F,double THRESH,
                      short pos_now);

//=====
===
double CalcCorrThresh(short * Hst,short HDim, short NPntAboveThr,
                      int PlusThresh, double PrcntLvl)
{double out;
 short sum=0,N=0,ist,incr;
 if (PrcntLvl)
   {for(ist=0;ist<HDim;N+=Hst[ist++]);
    NPntAboveThr=short (N*PrcntLvl+0.5);
   }

//-----
if(PlusThresh)
  {ist=HDim;incr=-1;}
else
  {ist=-1;incr=1;}
// POINT PARAMETER;
do{
  ist+=incr;
  sum+=*(Hst+ist);
}
while((sum<NPntAboveThr) && (ist>=0) && (ist<HDim));
out=ProbValue(HDim,ist);
return out;
}

```

```

//=====
double CorrelationEstim(double C, double MinVal, double MaxVal,
void*AddInf)
{
    if (C<MinVal) return MinVal;
    if (C>MaxVal) return MaxVal;
    return C;
}
//=====
COLOR_VEC (*PointColFunct())(COLOR_RGB p1, double Thresh_mdl)

{ switch ( MAP)
    {case NTSC:return(color_space_NTSC);
     case New_plan:return(color_space_NEW);
     case HSI:return(color_space_RGB);
     case RGB:return(color_space_RGB_simple);
     case LUMIN_THR:return(color_space_LUMIN_THR);
     case IHS:return(color_space_IHS);
    };
return NULL;
}
//=====
const short CH_HIGHT=80, CH_BASE=470;
//=====
void MatchForProtoStr(PCT &T, PRT &P, SCR_PNT winsize, short winstep,
double *CorThresh, RsltNow *NowRslt,
short *StripEnds)
{
    _setcolor( color_num(0,0,0));
    _rectangle( _GFILLINTERIOR, 0, CH_BASE-3*CH_HIGHT-(CH_HIGHT>>1),
vc.numxpixels ,CH_BASE );

    double crmdl;
    short i;
    char mess[40];
    short F=1;
    COLOR_VEC (*p_funct)(COLOR_RGB p1, double Thresh_mdl);
    p_funct=PointColFunct();
    // PCT win(winsize.c, winsize.r);
    // PCT tag(winsize.c, winsize.r);

    SCR_PNT st_t, st_win;
    AVERAGE_VEC middle_win, middle_tag;
    const AVERAGE_VEC z={{0,0,0},{0,0,0}};

    COLOR_VEC cr;

```

```

const COLOR_VEC z_col={0.0,0.0,0.0};

    st_t.r=0;
    st_win.r=0;
    short k,StripStart,StripStop;
    double ValueNow[1024];
    // double *ValueNow=new double[P.s_cols+T.s_cols];
    short PartNum;
    k=PartNum=0;
    while(StripEnds[PartNum]>=0)
        {StripStart=StripEnds[PartNum++];
        StripStop=StripEnds[PartNum++];
        for (st_win.c=StripStart;
        st_win.c+winsize.c<=StripStop;st_win.c+=winstep,k++)
        {
            F=1;
            middle_win=average2(P,Thresh_mdl,winsize,st_win,p_funct);

const COLOR_VEC z_UNIT={1.0,1.0,1.0};
            for(i=0;i<P.s_cols+T.s_cols;ValueNow[i++]=0.0);

            for (st_t.c=0;st_t.c<=T.s_cols-winsize.c;st_t.c++)
            {
short EndPointOfProNow= P.s_cols + st_t.c-st_win.c;
//=====
                middle_tag=average2(T,Thresh_mdl,winsize,st_t,p_funct);
// DIFF ABS VALUES
#ifdef SINGL_VAL
                cr=template_conv_2( T,P,Thresh_mdl,z_UNIT,winsize,st_t,st_win,
                p_funct);
//                strcpy(mess," VECTOR Approach to CORRELATION ");
                cr= Correlation_single_1(cr,middle_tag,middle_win,z_UNIT);
                crmdl=cr.c[0];
#else
#ifdef ABS_VALUE
//                strcpy(mess," DIFF ABS VALUES/ max ABS VALUES");
                cr=
                    template_abs_diff_1 (tag,win,Thresh_mdl,z_UNIT,p_funct,
                    middle_tag,middle_win);
#else
                cr=template_conv_1( tag,win,Thresh_mdl,z_UNIT,p_funct);
                strcpy(mess," PEARSON CORR. ");
                cr=Correlation(cr,middle_tag,middle_win,z_UNIT);
                crmdl=0.333333*(cr.c[0]+cr.c[1]+cr.c[2]);
#endif
#endif
            }

            if (crmdl> CorrThresh[k])

```

```

        ValueNow[EndPointOfProNow]=crmdl;

        draw_color_corr_1( cr, F,0.3,EndPointOfProNow);
        F=0;
    }
double old=0,next,Val,now;
next=ValueNow[0];
for(i=0;i<P.s_cols+T.s_cols;i++)
{
    now=next;
    next=(i==P.s_cols+T.s_cols-1)?0:ValueNow[i+1];
    Val=__max(__max(now,next),old);
    if(Val)
        {NowRslt[i].value+= Val;
         NowRslt[i].voices++;
        }
    old=now;
}
}
}
//===== delete (ValueNow);
return ;
}
//=====
=====
void draw_color_corr_1(COLOR_VEC corr,short F,
double THRESH,
short pos_now)
{
    short j,k,l,i,st;
    static short pos_old;
    short POS;
    static COLOR_RGB corr_old;
    POS=10+pos_now;
    _setcolor( color_num(240,240,240));
    if(F)
    {
        corr_old.r=k=CH_BASE-2*CH_HIGHT-40;
        st=CH_HIGHT/10;
        for(i=0;i<3;i++)
        {
            _moveto( 10,k-CH_HIGHT);
            _lineto(10,k);
            _lineto(10+vc.numxpixels,k);
            _moveto(10,k-CH_HIGHT*THRESH);
            _lineto(10+vc.numxpixels,k-CH_HIGHT*THRESH);
            for(l=0,j=1;j<11;j++)
                {l+=st;

```

```
        _moveto(
            (j==5)?5:((j==10)?0:7)
            ,k-l);
        _lineto(10,k-l);
    }
    k+=(CH_HIGHT+20);
}
corr_old.g=corr_old.r+CH_HIGHT+20;
corr_old.b=corr_old.g+CH_HIGHT+20;
pos_old=10;
}
_setcolor( color_num(240,240,240));
k=CH_BASE;
_moveto( pos_old,corr_old.b);
j=k-(short)(corr.c[2]*CH_HIGHT);
_lineto((short)(POS),j);
corr_old.b=j;

k+=(CH_HIGHT+20);
_moveto( pos_old,corr_old.g);
j=k-(short)(corr.c[1]*CH_HIGHT);
_lineto((short)(POS),j);
corr_old.g=j;

k+=(CH_HIGHT+20);
_moveto( pos_old,corr_old.r);
j=k-(short)(corr.c[0]*CH_HIGHT);
_lineto((short)(POS),j);
corr_old.r=j;
pos_old=POS;
}
```

```

#include <stdlib.h>
#include <conio.h>
#include <stdio.h>
#include <string.h>
#include <graph.h>
#include <float.h>
#include <time.h>
#include <ctype.h>
#include <fstream.h>
#include <afx.h>

#include "matchng.h"
#include "PIC_PRO.h"
#include "MtchTpl2.h"
#include "shift.h"
#include "filemnp.h"

#ifdef _DEBUG
#define new DEBUG_NEW
#endif

//input Par
//Files <frame>.RGB; <frame>.sgs; <proto's>.RGB; <proto's>.hs2
short PlusThresnPnt; //Threshold for histogramme Number Of Points
double MinimalVal; // for CorrThresh
double MaximalVal;
short HighVoiceThreshold=6;
short median=0;

short MAP=1; // Color Space

short WinNum;
double CorrThresh[MaxWinNum];
int Introduction(int arg,char *a);
short NumberOfWin,HistDim;
CString BestName[2]={CString::CString(30),CString::CString(30)};

#define MaxNumberOfPoints 1024
RsItNow NowRsIt[MaxNumberOfPoints];

short ReadHist(const char *PName,short ***H);
const short NumberOfShifts=sizeof(VShift)/(4*sizeof(double));

void FreeHist( short NOfWin,short **H);
void draw_DMF(RsItNow Now ,short F,short CH_HEIGHT,short CH_BASE,
short pos_now,double scale);

double GAMMA=1.0,CORR_THRESH=0.0,Thresh_mdl=0.0;
short VOITING=3,TAG_hight;

```



```

struct _videoconfig vc;
ofstream LineC;
const char clean[]=""
//=====
=====
int ReadStrInf(char *name,short *StD)
{ifstream InpF;
char a[80];
strcat(strcpy(a,name),".str");
short i;
InpF.open(a,ios::in|ios::nocreate);
if(InpF.fail())
{InpF.clear(0);
return 1;
}
InpF.getline(a,80,'\n');
InpF.getline(a,80,'\n');
InpF.getline(a,80,'\n');
InpF.getline(a,80,'\n');
InpF.getline(a,80,'\n');
InpF.getline(a,80,'\n');
i=0;
do
{
InpF>>StD[i++];
if (InpF.eof()|| i>17)
{ StD[-i]=-1;
break;
}
InpF>>StD[i++];
}
while(1);
InpF.close();
return 0;
}
//=====
void PUT_NO_MATCH(void)
{
_settextposition( vc.numtextrows-2,0);
_outtext( clean );
_settextposition( vc.numtextrows-2,0);
_outtext( "NO MATCH" );
}
//=====
=====
void FillProtocol(char *m,short StrN,const char *name, RSLT now, int
AskMess)
{char mess[40],*p;
LineC<< StrN<<" \t"<<name<<" \t"<<now.ShNumb<<" \t"<<now.pos<<" \t"<<
now.R.voices<<" \t"<<now.R.value<<" \t";

```

```

if (AskMess)
{
    if(now.R.value)
    {
        _settextposition( vc.numtextrows-2,0);
        _outtext( clean );
        _settextposition( vc.numtextrows-2,0);
        sprintf(mess," %s V=%6g N=%2d",m,now.R.value,now.R.voices);
        _outtext(mess );
        p=mess;
        while(!isspace( (int)(*p++=(char)getche()) ));
        *(-p)='\0';
        LineC<<" \t"<<mess;
    }
    else
        LineC<<" \t NO MATCH";
}
LineC<<"\n";
}
//=====
=====
void PresentRslt(char *m,PRT &T,RSLT &R,SCR_PNT *V, short hight,short
row_n,
                char *p_msk,short CL)
{
    SCR_PNT p_p(10,row_n);
    if(R.R.value)
    {
        if(CL)
        {
            _setcolor( color_num(0,0,0));
            _rectangle( _GFillInterior,0,p_p.r,
                        680,p_p.r+36);
        }
        _setcolor( color_num(240,240,240));
        PCT t_map=linear_transform(T,V,hight);
        //PCT t_map=linear_transform_cont(T,V,hight);
        sign_present_RGB( t_map,p_p);
        t_map.free_PCT;
        p_p.r+=20;
        char p_name[40];
        PRT P;
        next_pict(p_name,p_msk,".rgb",R.ProtoNum);
        if(P.proto_storage_rgb(p_name,vc,".str"))
            {printf("RGB PROTO not exist"); GRAPH_OUT(-1);
            };
        p_p.c=10+R.pos-P.s_cols;
        sign_present_RGB( P,p_p);
        FillProtocol(m,R.StrNum,(const char*) P.SIGN_name,
                    R,TRUE);
    }
}

```

```

P.free_PCT;
}
else
{
    PUT_NO_MATCH();
}
}
//=====
=====
CString PROTOCOL_NAME;

//===== OpenLineCollection
void OpenLineCol(const char*name)
{ LineC.open(name,ios::out|ios::app|ios::nocreate);
  if(LineC.fail())
  {LineC.clear(0);
   LineC.open(name,ios::out|ios::app|ios::noreplace);
   if(LineC.fail())
   {LineC.clear(0);
    cout << "CAN NOT OPEN FILE "<<name;
    GRAPH_OUT(-1);
   }
  }
}
//=====
=====
int main(int argc,char* argv[])
{
    short **Hist;
    const int PlusDir= TRUE;
    short StrDescr[17]; // ONLY 6 partition for string

    if(Introduction(argc,argv[argc-1])) return-1;

    short CalcHight=MaskDecoding(argv[2]); // PROTOTYPE HIGHT

    // ===== GRAPHICS START
    if(GRAPHICS_START(&vc,GRAPHMODE)) GRAPH_OUT(-1);
    PROTOCOL_NAME=argv[1];
    PROTOCOL_NAME+="mch";
    OpenLineCol((const char*)PROTOCOL_NAME);

    //===== TARGET LOADING & MEMORY PARAMETERS
    DEFINITION
    PRT TARGET;
    PRT PROTO;
    PCT target_map;
    const SCR_PNT proto_pos(10,30),target_pos(10,10);

```

```

SCR_PNT
TARGET_vrt[STR_MAX][4],NewVrt[STR_MAX][NumberOfShifts][4];
if(TARGET.proto_storage_rgb(argv[1],vc))
{printf("RGB TARGET not exist");    GRAPH_OUT(-1);
return -1;
}
int NumberOfStrips=TARGET.read_target_SGN(TARGET_vrt,EXTANTION);
LineC<< " MAP \tPlusThresnPnt \tMinimalVal \tMaximalVal \n";
LineC<<MAP<<" \t"<<PlusThresnPnt<<" \t"<<MinimalVal
<<" \t"<<MaximalVal<<" \t"<<median<<"\n";
LineC<< "File \t"<< argv[1]<<" \t "<<TARGET.SIGN_name<<"\n";
LineC<< "S# \t PROTO \t Sh \t Pos \t V \t Value \t Res\n";

if(!NumberOfStrips)
{LineC<<"Did NOT find strips\n";
LineC.close();
printf("Did not find lines"); GRAPH_OUT(-1);
return -1;
}

char proto_name[40];
short ProtoNumber=0;

RSLT BestRsIts[2];
BestRsIts[0]=RSLT::RSLT(0,0,0,-1,0,0);
BestRsIts[1]=RSLT::RSLT(0,0,0,-1,0,0);
RSLT BestNowRsIts[2];

//      New Vertexes Calculation
short StripNumber;
short ShiftNumber;

for(StripNumber=0;StripNumber<NumberOfStrips;StripNumber++)
for(ShiftNumber=0;ShiftNumber<NumberOfShifts;ShiftNumber++) //
return NewVrt after reordering
VrtxCalculation( TARGET,VShift[ShiftNumber],
NewVrt[StripNumber][ShiftNumber],
TARGET_vrt[StripNumber]);

//      LOOP OVER STRIPS
for(StripNumber=0;StripNumber<NumberOfStrips;StripNumber++)
{
//      LOOP OVER PROTOTYPE
short ProtoNumber=0;
while(next_pict(proto_name,argv[2],".rgb",ProtoNumber)) // RGB
proto EXIST
{
//      LOCAL QUALITY
BestNowRsIts[0]=RSLT::RSLT(0,0,0,-1,0,0);
BestNowRsIts[1]=RSLT::RSLT(0,0,0,-1,0,0);

```

```

//=====Proto Loading
    if(PROTO.proto_storage_rgb(proto_name,vc,".str"))
        {printf("RGB PROTO not exist"); GRAPH_OUT(-1);return -1;
        };
    _clearscreen( _GCLEARSCREEN );
    sign_present_RGB( PROTO,proto_pos);
    if(ReadStrInf(proto_name,StrDescr))
        {printf("SGN PROTO not exist"); GRAPH_OUT(-1);return -1;
        };

//===== HIST reading
CString HName("-",60);
    HName=proto_name;
    HName+=".hs2";
// here read size of windows from Hist and so on
SCR_PNT WinSize;
short winstep,NoHist;
    WinSize.r=CalcHight;
    NoHist=0;
    if(!(WinSize.c=ReadHist((const char *)HName,&Hist)))
        {printf("Did not find Hist %s",proto_name);
        NoHist=1;
        WinSize.c=8;
        NumberOfWin=MaxWinNum;}
    winstep=WinSize.c;

//-----MEMORY ALLOCATION & initialisation FOR Decision Making
short i;
    for (i=0;i<MaxWinNum;CorrThresh[i++]=-1.0);
    for (i=0;i<NumberOfWin;i++)
        {if(NoHist)
            CorrThresh[i]=0.3;
        else
            {
                CorrThresh[i]=CalcCorrThresh(*(Hist+i),HistDim,
                    PlusThreshnPnt,PlusDir,0);
                CorrThresh[i]=CorrelationEstim(CorrThresh[i],
                    MinimalVal,
                        MaximalVal,NULL);
            }
        }
    if(!NoHist)
        FreeHist( NumberOfWin, Hist);
// !!!!!!! CorrelationThreshold for all windows
short NumberOfPoints;
//===== LOOP OVER SHIFTS
    for(ShiftNumber=0;ShiftNumber<NumberOfShifts;
        ShiftNumber++)
        {if(NewVrt[StripNumber][ShiftNumber][0].r<0) continue;

```

```

        target_map=linear_transform(TARGET,
//      target_map=linear_transform_cont(TARGET,
        NewVrt[StripNumber][ShiftNumber],CalcHight );
        NumberOfPoints=PROTO.s_cols+target_map.s_cols;
        _setcolor( color_num(0,0,0));
        _rectangle( _GFILLINTERIOR,0,0,
        vc.numxpixels ,proto_pos.r-1);

        sign_present_RGB( target_map,target_pos);

//--- Result Initialisation For Every SHIFT
        for(i=0;i<MaxNumberOfPoints;i++)
            NowRslt[i]=RsltNow::RsltNow();
//===== Proto Calculation
        MatchForProtoStr(target_map, PROTO, WinSize, winstep,
        CorrThresh,NowRslt,StrDescr);
        target_map.free_PCT();

//correct filling The BEST For Shift,Proto,Strip
#ifdef MYDEBUG
double scale=-1;
for(i=0;i<NumberOfPoints;i++)
    scale=__max(scale, NowRslt[i].value);
#endif
for(i=0;i<NumberOfPoints;i++)
{
    if( NowRslt[i].value>BestNowRsIts[0].R.value)
    {
//===== Not repeat small shifts
        if(abs(i-BestNowRsIts[0].pos)>=4)
            BestNowRsIts[1]=BestNowRsIts[0];
        BestNowRsIts[0].pos=i;
        BestNowRsIts[0].R=NowRslt[i];
        BestNowRsIts[0].ShNumb=ShiftNumber;
        BestNowRsIts[0].ProtoNum=ProtoNumber;
        BestNowRsIts[0].StrNum=StripNumber;
        if(BestNowRsIts[0].R.voices>=HighVoiceThreshold)
        {BestRsIts[0]=BestNowRsIts[0];
        BestRsIts[1]=RSLT::RSLT(0,0,0,-1,0,0);
        PROTO.free_PCT();
        BestName[0]=PROTO.SIGN_name;
        goto DIRECT_DECISION;
        }
    }
    else
        if( (NowRslt[i].value>BestNowRsIts[1].R.value) &&
        (abs(i-BestNowRsIts[0].pos)>=4))
        {
            BestNowRsIts[1].pos=i;

```

```

        BestNowRslts[1].R=NowRslt[i];
        BestNowRslts[1].ShNumb=ShiftNumber;
        BestNowRslts[1].ProtoNum=ProtoNumber;
        BestNowRslts[1].StrNum=StripNumber;
    }
}
//_____ May be presentation for analyses
#ifdef MYDEBUG
short F=1;
short CH_HEIGHT=100, CH_BASE=190;

_setcolor( color_num(0,0,0));
_rectangle( _GFillInterior,0,CH_BASE-CH_HEIGHT,
            vc.numxpixels ,CH_BASE );
    if(scale)
    {
        for(i=0;i<NumberOfPoints;i++)
            {draw_DMF(NowRslt[i],F, CH_HEIGHT, CH_BASE,
                      i-PROTO.s_cols, scale);
              F=0;
            }
    }
#endif RUN
    getch();
#endif
    }
    else
        PUT_NO_MATCH();
#endif
    } // Shift closed
    PROTO.free_PCT();
    ProtoNumber++;
//=====PROTOCOL PROTO
OUTPUT=====
#ifdef MYDEBUG
short pp2=10,CL2=TRUE;
for (i=0;i<2;i++)
    {
        PresentRslt("PROTO SHIFT RSLT?",TARGET,BestNowRslts[i],
                    NewVrt[BestNowRslts[i].StrNum][BestNowRslts[i].ShNumb],
                    CalcHight,pp2,argv[2],CL2);

        CL2=FALSE;
        pp2=50;
    }
#else
    FillProtocol("PROTO SHIFT RSLT?",StripNumber,(const char*)
PROTO.SIGN_name,
                BestNowRslts[0],FALSE);
#endif

```

```

//      GLOBAL MAX
for(i=0;i<2;i++)
    if( BestNowRslts[i].R.value>BestRslts[0].R.value)
        {BestRslts[1]=BestRslts[0];
          BestRslts[0]=BestNowRslts[i];
          BestName[1]=BestName[0];
          BestName[0]=PROTO.SIGN_name;
        }
    else
        if( BestNowRslts[i].R.value>BestRslts[1].R.value)
            {BestRslts[1]=BestNowRslts[i];
              BestName[1]=PROTO.SIGN_name;
            }

//-----
    } //Proto closed
    } //Strips Closed
//      GLOBAL PROTOCOL OUTPUT
DIRECT_DECISION:
#ifdef MYDEBUG
    short i;
    short Rows_n=10,Blanc=TRUE;
    LineC<<" Result.....\n";
    _clearscreen( _GCLEARSCREEN );
    for (i=0;i<2;i++)
        {
            PresentRslt("GLOBAL RSLT
?",TARGET,BestRslts[i],NewVrt[BestRslts[i].StrNum][BestRslts[i].ShNumb],
                        CalcHight,Rows_n,argv[2],Blanc);
            Blanc=FALSE;
            Rows_n+=50;
        }
#else
    FillProtocol("GLOBAL RSLT ?",(const char *)BestRslts[0].StrNum,
                (const char*) BestName[0];
                BestRslts[0],FALSE);

#endif
//
LineC.close();
TARGET.free_PCT();
GRAPH_OUT(0);
return 0;
}
//=====
=
short ReadHist(const char *PName,short ***H)
{char p[80];
 ifstream HFile;
 HFile.open(PName,ios::in|ios::nocreate);

```



```

    if(HFile.fail())
    {HFile.clear(0);
      cout << "CAN NOT OPEN FILE "<<PName<<"\n";
      return 0;
    }

    HFile.getline(p,80); // LineC<<"Histogrammes\n";
    HFile.getline(p,80); //<<argv[1]<<"\t"<<prototype.SIGN_name<<"\n";
    HFile>>p>>NumberOfWin; // <<NumberOfWin<<"\n";

    HFile>>p>>HistDim; // "NumberOfBins-1\t"<<HistDim<<"\n";
    HFile.getline(p,80); // "Win_pos\n";
    HFile.getline(p,80); // "Win_pos\n";

    (*H)= new (short(*[NumberOfWin]));
    short n,j,i,WindowSize;
    for (i=0;i<NumberOfWin;i++)
      (*H)[i]=new short[HistDim];
    for(j=0;j<NumberOfWin;j++)
      {if(j==1)
        HFile>>WindowSize;
        else
          HFile>>n;
        for(i=0;i<HistDim;i++)
          HFile>>(*H)[j][i];
      }
    HFile.close();
    return(WindowSize);
  }
  //=====
  void FreeHist( short NOfWin, short **H)
  {short i;
    for (i=0;i<NOfWin;i++)
      delete H[i];
    delete H;
  }

  //
  //=====
  int get_number_match() // INITIALISATION GRAPHICMODE, GET SCALE
  {
    GRAPH_OUT();
    cout << " MAP PlusThresnPnt MinimalVal MaximalVal
    HighVoiceThreshold median \n";
  }

```

```

        cin
>>MAP>>PlusThresnPnt>>MinimalVal>>MaximalVal>>HighVoiceThreshold
>>median;

// ===== GRAPHICS START
    if(GRAPHICS_START(&vc,GRAPHMODE)) GRAPH_OUT(-1);
//=====
return 1;
}
//=====
int get_number_match_f(FILE *f) // INITIALISATION GRAPHICMODE, GET
SCALE
{
    fscanf(f," %*[^\\n]s ");
    fscanf(f," %d %d %lf %lf %d
%d",&MAP,&PlusThresnPnt,&MinimalVal,&MaximalVal,
        &HighVoiceThreshold,&median);
    // Threshold for histogramme    Number Of Points
    // for CorrThresh
return 1;
}
//=====INTRODUCTION
int Introduction(int arg, char *a)
{
    int FLG_F=0;
    FILE *datainf;
    short n=0;
    if((arg != 3) && (arg !=4))
    {
        printf(" target-file proto_file_mask \\n");
        FLG_F=0;
        return(1);
    }
    else
        if(arg ==4)
        {FLG_F=1;
        if(!(datainf=fopen(a,"r"))) return 0;
        }
    if(FLG_F)
        {get_number_match_f(datainf); //    GET SCALE AND PARAMETERS
        fclose (datainf);
        }
    else
        get_number_match();
return 0;
}
//=====
//=====
//----- Graphic Output

```

```

//=====
=====
void draw_DMF(RsltNow Now ,short F,short CH_HIGHT,
             short CH_BASE,
             short pos_now,double scale)
{
    short j,k,l,st;
    static short real_size,pos_old;
    short POS;
    static double old_Y;
    POS=10+pos_now;
    _setcolor( color_num(240,240,240));
    if(F)
    {
        old_Y=k=CH_BASE;
        st=CH_HIGHT/10;
        _moveto( 10,k-CH_HIGHT);
        _lineto(10,k);
        _lineto(680,k);
        _moveto(10,k-CH_HIGHT);
        _lineto(680,k-CH_HIGHT);
        for(l=0,j=1;j<11;j++)
        {
            l+=st;
            _moveto(
                (j==5)?5:((j==10)?0:7)
                ,k-l);
            _lineto(10,k-l);
        }
        pos_old=10;
    }
    _moveto( pos_old,old_Y);
    j=CH_BASE-(short)(Now.value*CH_HIGHT/scale);
    _lineto((short)(POS),j);
    old_Y=j;
    pos_old=POS;
}
//=====
=====

```

```

ORIGIN = PWB
ORIGIN_VER = 2.0
PROJ = STRNEW
PROJFILE = STRNEW.MAK
BUILDDIR = obj
DEBUG = 1

```

```

BRFLAGS = /o obj\$(PROJ).bsc
BSCMAKE = bscmake
SBRPACK = sbrpack
NMAKEBSC1 = set
NMAKEBSC2 = nmake
BROWSE = 1
CC = cl
CFLAGS_G = /W2 /BATCH /FR$*.sbr /Zn
CFLAGS_D = /f /Zi /Od
CFLAGS_R = /f- /Ot /Oi /OI /Oe /Og /Gs
CXX = cl
CXXFLAGS_G = /AL /W4 /G2 /D_DOS /BATCH /FR$*.sbr /Zn
CXXFLAGS_D = /f- /Ob1 /Od /FPi87 /Zi /DMYDEBUG /DRUN /D_DEBUG
CXXFLAGS_R = /f- /Os /OI /Og /Oe /Oi /FPi87 /Gs /DMYDEBUG /DRUN
MAPFILE_D = NUL
MAPFILE_R = NUL
LFLAGS_G = /NOI /STACK:32000 /BATCH /ONERROR:NOEXE
LFLAGS_D = /CO /FAR /PACKC
LFLAGS_R = /EXE /FAR /PACKC
LINKER = link
ILINK = ilink
LRF = echo > NUL
ILFLAGS = /a /e
LLIBS_R = LAFXCR
LLIBS_D = LAFXCRD
LLIBS_G = graphics
CVFLAGS = /25 /S
RUNFLAGS = TST\t10284o plnew.ini

```

```

FILES = COR_FNC8.CPP PIC_M8.CPP VICAL8.CPP RES_MCH8.CPP
PROJCTN8.CPP\
COMP_FNC.CPP STRNEW.CPP
OBJS = obj\COR_FNC8.obj obj\PIC_M8.obj obj\VICAL8.obj
obj\RES_MCH8.obj\
obj\PROJCTN8.obj obj\COMP_FNC.obj obj\STRNEW.obj
SBRS = obj\COR_FNC8.sbr obj\PIC_M8.sbr obj\VICAL8.sbr
obj\RES_MCH8.sbr\
obj\PROJCTN8.sbr obj\COMP_FNC.sbr obj\STRNEW.sbr

```

```
all: obj\$(PROJ).exe
```

```
.SUFFIXES:
```

.SUFFIXES:

.SUFFIXES: .obj .sbr .cpp

```
obj\COR_FNC8.obj : COR_FNC8.CPP C:\C700\INCLUDE\stdio.h
    C:\C700\INCLUDE\stdlib.h C:\C700\INCLUDE\conio.h
    C:\C700\INCLUDE\float.h C:\C700\INCLUDE\graph.h cor_fnc8.h
projctn8.h\
    pic_mch8.h res_mch8.h C:\C700\INCLUDE\math.h
C:\C700\INCLUDE\vmemory.h
!IF $(DEBUG)
    @$(CXX) @<<obj\$(PROJ).rsp
/c $(CXXFLAGS_G)
$(CXXFLAGS_D) /Foobj\COR_FNC8.obj COR_FNC8.CPP
<<
!ELSE
    @$(CXX) @<<obj\$(PROJ).rsp
/c $(CXXFLAGS_G)
$(CXXFLAGS_R) /Foobj\COR_FNC8.obj COR_FNC8.CPP
<<
!ENDIF
```

```
obj\COR_FNC8.sbr : COR_FNC8.CPP C:\C700\INCLUDE\stdio.h
    C:\C700\INCLUDE\stdlib.h C:\C700\INCLUDE\conio.h
    C:\C700\INCLUDE\float.h C:\C700\INCLUDE\graph.h cor_fnc8.h
projctn8.h\
    pic_mch8.h res_mch8.h C:\C700\INCLUDE\math.h
C:\C700\INCLUDE\vmemory.h
!IF $(DEBUG)
    @$(CXX) @<<obj\$(PROJ).rsp
/Zs $(CXXFLAGS_G)
$(CXXFLAGS_D) /FRobj\COR_FNC8.sbr COR_FNC8.CPP
<<
!ELSE
    @$(CXX) @<<obj\$(PROJ).rsp
/Zs $(CXXFLAGS_G)
$(CXXFLAGS_R) /FRobj\COR_FNC8.sbr COR_FNC8.CPP
<<
!ENDIF
```

```
obj\PIC_M8.obj : PIC_M8.CPP C:\C700\INCLUDE\stdlib.h
C:\C700\INCLUDE\stdio.h
    C:\C700\INCLUDE\graph.h C:\C700\INCLUDE\math.h
C:\C700\INCLUDE\io.h\
    C:\C700\INCLUDE\fcntl.h C:\C700\INCLUDE\string.h\
    C:\C700\INCLUDE\float.h C:\C700\INCLUDE\malloc.h phdr.h vical8.h\
    pic_mch8.h C:\C700\INCLUDE\vmemory.h projctn8.h
!IF $(DEBUG)
    @$(CXX) @<<obj\$(PROJ).rsp
/c $(CXXFLAGS_G)
```

```
$(CXXFLAGS_D) /Foobj\PIC_M8.obj PIC_M8.CPP
<<
```

```
!ELSE
```

```
    @$(CXX) @<<obj\$(PROJ).rsp
/c $(CXXFLAGS_G)
$(CXXFLAGS_R) /Foobj\PIC_M8.obj PIC_M8.CPP
<<
```

```
!ENDIF
```

```
obj\PIC_M8.sbr : PIC_M8.CPP C:\C700\INCLUDE\stdlib.h
```

```
C:\C700\INCLUDE\stdio.h\
```

```
    C:\C700\INCLUDE\graph.h C:\C700\INCLUDE\math.h
```

```
C:\C700\INCLUDE\io.h\
```

```
    C:\C700\INCLUDE\fcntl.h C:\C700\INCLUDE\string.h\
```

```
    C:\C700\INCLUDE\float.h C:\C700\INCLUDE\malloc.h phdr.h vical8.h\
```

```
    pic_mch8.h C:\C700\INCLUDE\vmemory.h projctn8.h
```

```
!IF $(DEBUG)
```

```
    @$(CXX) @<<obj\$(PROJ).rsp
/Zs $(CXXFLAGS_G)
$(CXXFLAGS_D) /FRobj\PIC_M8.sbr PIC_M8.CPP
<<
```

```
!ELSE
```

```
    @$(CXX) @<<obj\$(PROJ).rsp
/Zs $(CXXFLAGS_G)
$(CXXFLAGS_R) /FRobj\PIC_M8.sbr PIC_M8.CPP
<<
```

```
!ENDIF
```

```
obj\VICAL8.obj : VICAL8.CPP C:\C700\INCLUDE\stdlib.h
```

```
C:\C700\INCLUDE\stdio.h\
```

```
    C:\C700\INCLUDE\vmemory.h C:\C700\INCLUDE\malloc.h
```

```
!IF $(DEBUG)
```

```
    @$(CXX) @<<obj\$(PROJ).rsp
/c $(CXXFLAGS_G)
$(CXXFLAGS_D) /Foobj\VICAL8.obj VICAL8.CPP
<<
```

```
!ELSE
```

```
    @$(CXX) @<<obj\$(PROJ).rsp
/c $(CXXFLAGS_G)
$(CXXFLAGS_R) /Foobj\VICAL8.obj VICAL8.CPP
<<
```

```
!ENDIF
```

```
obj\VICAL8.sbr : VICAL8.CPP C:\C700\INCLUDE\stdlib.h
```

```
C:\C700\INCLUDE\stdio.h\
```

```
    C:\C700\INCLUDE\vmemory.h C:\C700\INCLUDE\malloc.h
```

```
!IF $(DEBUG)
```

```
    @$(CXX) @<<obj\$(PROJ).rsp
/Zs $(CXXFLAGS_G)
```

```

$(CXXFLAGS_D) /FRobj\VICAL8.sbr VICAL8.CPP
<<
!ELSE
    @$(CXX) @<<obj\$(PROJ).rsp
/Zs $(CXXFLAGS_G)
$(CXXFLAGS_R) /FRobj\VICAL8.sbr VICAL8.CPP
<<
!ENDIF

obj\RES_MCH8.obj : RES_MCH8.CPP C:\C700\INCLUDE\stdlib.h\
    C:\C700\INCLUDE\memory.h res_mch8.h
C:\C700\INCLUDE\graph.h\
    projctn8.h pic_mch8.h C:\C700\INCLUDE\math.h
!IF $(DEBUG)
    @$(CXX) @<<obj\$(PROJ).rsp
/c $(CXXFLAGS_G)
$(CXXFLAGS_D) /Foobj\RES_MCH8.obj RES_MCH8.CPP
<<
!ELSE
    @$(CXX) @<<obj\$(PROJ).rsp
/c $(CXXFLAGS_G)
$(CXXFLAGS_R) /Foobj\RES_MCH8.obj RES_MCH8.CPP
<<
!ENDIF

obj\RES_MCH8.sbr : RES_MCH8.CPP C:\C700\INCLUDE\stdlib.h\
    C:\C700\INCLUDE\memory.h res_mch8.h
C:\C700\INCLUDE\graph.h\
    projctn8.h pic_mch8.h C:\C700\INCLUDE\math.h
!IF $(DEBUG)
    @$(CXX) @<<obj\$(PROJ).rsp
/Zs $(CXXFLAGS_G)
$(CXXFLAGS_D) /FRobj\RES_MCH8.sbr RES_MCH8.CPP
<<
!ELSE
    @$(CXX) @<<obj\$(PROJ).rsp
/Zs $(CXXFLAGS_G)
$(CXXFLAGS_R) /FRobj\RES_MCH8.sbr RES_MCH8.CPP
<<
!ENDIF

obj\PROJECTN8.obj : PROJECTN8.CPP C:\C700\INCLUDE\graph.h\
    C:\C700\INCLUDE\stdlib.h C:\C700\INCLUDE\iostream.h projctn8.h\
    C:\C700\INCLUDE\ios.h C:\C700\INCLUDE\streamb.h\
    C:\C700\INCLUDE\istream.h C:\C700\INCLUDE\ostream.h\
    C:\C700\INCLUDE\math.h
!IF $(DEBUG)
    @$(CXX) @<<obj\$(PROJ).rsp
/c $(CXXFLAGS_G)

```

```
$(CXXFLAGS_D) /Foobj\PROJECTN8.obj PROJECTN8.CPP
<<
```

```
!ELSE
```

```
    @$(CXX) @<<obj\$(PROJ).rsp
/c $(CXXFLAGS_G)
$(CXXFLAGS_R) /Foobj\PROJECTN8.obj PROJECTN8.CPP
<<
```

```
!ENDIF
```

```
obj\PROJECTN8.sbr : PROJECTN8.CPP C:\C700\INCLUDE\graph.h
    C:\C700\INCLUDE\stdlib.h C:\C700\INCLUDE\iostream.h projctn8.h\
    C:\C700\INCLUDE\ios.h C:\C700\INCLUDE\streamb.h\
    C:\C700\INCLUDE\istream.h C:\C700\INCLUDE\ostream.h\
    C:\C700\INCLUDE\math.h
```

```
!IF $(DEBUG)
```

```
    @$(CXX) @<<obj\$(PROJ).rsp
/Zs $(CXXFLAGS_G)
$(CXXFLAGS_D) /FRobj\PROJECTN8.sbr PROJECTN8.CPP
<<
```

```
!ELSE
```

```
    @$(CXX) @<<obj\$(PROJ).rsp
/Zs $(CXXFLAGS_G)
$(CXXFLAGS_R) /FRobj\PROJECTN8.sbr PROJECTN8.CPP
<<
```

```
!ENDIF
```

```
obj\COMP_FNC.obj : COMP_FNC.CPP comp_fnc.h projctn8.h
    C:\C700\INCLUDE\math.h
    C:\C700\INCLUDE\graph.h
```

```
!IF $(DEBUG)
```

```
    @$(CXX) @<<obj\$(PROJ).rsp
/c $(CXXFLAGS_G)
$(CXXFLAGS_D) /Foobj\COMP_FNC.obj COMP_FNC.CPP
<<
```

```
!ELSE
```

```
    @$(CXX) @<<obj\$(PROJ).rsp
/c $(CXXFLAGS_G)
$(CXXFLAGS_R) /Foobj\COMP_FNC.obj COMP_FNC.CPP
<<
```

```
!ENDIF
```

```
obj\COMP_FNC.sbr : COMP_FNC.CPP comp_fnc.h projctn8.h
    C:\C700\INCLUDE\math.h
    C:\C700\INCLUDE\graph.h
```

```
!IF $(DEBUG)
```

```
    @$(CXX) @<<obj\$(PROJ).rsp
/Zs $(CXXFLAGS_G)
$(CXXFLAGS_D) /FRobj\COMP_FNC.sbr COMP_FNC.CPP
<<
```



```

ELSE
    @$(CXX) @<<obj$(PROJ).rsp
/Zs $(CXXFLAGS_G)
$(CXXFLAGS_R) /FRobj\COMP_FNC.sbr COMP_FNC.CPP
<<
!ENDIF

obj\STRNEW.obj : STRNEW.CPP C:\C700\INCLUDE\stdlib.h
C:\C700\INCLUDE\conio.h
C:\C700\INCLUDE\stdio.h C:\C700\INCLUDE\string.h\
C:\C700\INCLUDE\graph.h C:\C700\INCLUDE\float.h\
C:\C700\MFC\INCLUDE\afx.h C:\C700\INCLUDE\fstream.h\
C:\C700\INCLUDE\time.h mylibmd.h comp_fnc.h cor_fnc8.h\
C:\C700\INCLUDE\ctype.h C:\C700\MFC\INCLUDE\afx.inl\
C:\C700\INCLUDE\iostream.h projctn8.h pic_mch8.h res_mch8.h\
C:\C700\INCLUDE\ios.h C:\C700\INCLUDE\streamb.h\
C:\C700\INCLUDE\istream.h C:\C700\INCLUDE\ostream.h\
C:\C700\INCLUDE\math.h C:\C700\INCLUDE\vmemory.h

!IF $(DEBUG)
    @$(CXX) @<<obj$(PROJ).rsp
/c $(CXXFLAGS_G)
$(CXXFLAGS_D) /Foobj\STRNEW.obj STRNEW.CPP
<<
!ELSE
    @$(CXX) @<<obj$(PROJ).rsp
/c $(CXXFLAGS_G)
$(CXXFLAGS_R) /Foobj\STRNEW.obj STRNEW.CPP
<<
!ENDIF

obj\STRNEW.sbr : STRNEW.CPP C:\C700\INCLUDE\stdlib.h
C:\C700\INCLUDE\conio.h
C:\C700\INCLUDE\stdio.h C:\C700\INCLUDE\string.h\
C:\C700\INCLUDE\graph.h C:\C700\INCLUDE\float.h\
C:\C700\MFC\INCLUDE\afx.h C:\C700\INCLUDE\fstream.h\
C:\C700\INCLUDE\time.h mylibmd.h comp_fnc.h cor_fnc8.h\
C:\C700\INCLUDE\ctype.h C:\C700\MFC\INCLUDE\afx.inl\
C:\C700\INCLUDE\iostream.h projctn8.h pic_mch8.h res_mch8.h\
C:\C700\INCLUDE\ios.h C:\C700\INCLUDE\streamb.h\
C:\C700\INCLUDE\istream.h C:\C700\INCLUDE\ostream.h\
C:\C700\INCLUDE\math.h C:\C700\INCLUDE\vmemory.h

!IF $(DEBUG)
    @$(CXX) @<<obj$(PROJ).rsp
/Zs $(CXXFLAGS_G)
$(CXXFLAGS_D) /FRobj\STRNEW.sbr STRNEW.CPP
<<
!ELSE
    @$(CXX) @<<obj$(PROJ).rsp
/Zs $(CXXFLAGS_G)

```

```
$(CXXFLAGS_R) /FRobj\STRNEW.sbr STRNEW.CPP
<<
!ENDIF
```

```
obj\$(PROJ).bsc : $(SBRs)
    $(BSCMAKE) @<<
$(BRFLAGS) $(SBRs)
<<
```

```
obj\$(PROJ).exe : $(OBJS)
    -$(NMAKEBSC1) MAKEFLAGS=
    -$(NMAKEBSC2) $(NMFLAGS) -f $(PROJFILE) obj\$(PROJ).bsc
!IF $(DEBUG)
    $(LRF) @<<obj\$(PROJ).lrf
$(RT_OBJS: = +^
) $(OBJS: = +^
)
$@
$(MAPFILE_D)
$(LIBS: = +^
) +
$(LLIBS_G: = +^
) +
$(LLIBS_D: = +^
)
$(DEF_FILE) $(LFLAGS_G) $(LFLAGS_D);
<<
!ELSE
    $(LRF) @<<obj\$(PROJ).lrf
$(RT_OBJS: = +^
) $(OBJS: = +^
)
$@
$(MAPFILE_R)
$(LIBS: = +^
) +
$(LLIBS_G: = +^
) +
$(LLIBS_R: = +^
)
$(DEF_FILE) $(LFLAGS_G) $(LFLAGS_R);
<<
!ENDIF
$(LINKER) @obj\$(PROJ).lrf
```

```
.cpp.obj:
!IF $(DEBUG)
```

```
        @$(CXX) @<<obj\$(PROJ).rsp
/c $(CXXFLAGS_G)
$(CXXFLAGS_D) /Fo$@ $<
<<
!ELSE
        @$(CXX) @<<obj\$(PROJ).rsp
/c $(CXXFLAGS_G)
$(CXXFLAGS_R) /Fo$@ $<
<<
!ENDIF
```

```
.cpp.sbr :
!IF $(DEBUG)
        @$(CXX) @<<obj\$(PROJ).rsp
/Zs $(CXXFLAGS_G)
$(CXXFLAGS_D) /FR$@ $<
<<
!ELSE
        @$(CXX) @<<obj\$(PROJ).rsp
/Zs $(CXXFLAGS_G)
$(CXXFLAGS_R) /FR$@ $<
<<
!ENDIF
```

```
run: obj\$(PROJ).exe
      obj\$(PROJ).exe $(RUNFLAGS)
```

```
debug: obj\$(PROJ).exe
       CV $(CVFLAGS) obj\$(PROJ).exe $(RUNFLAGS)
```

```
//      LINE SELECTION
// COMMAND STRING
//
// Ins_corr <TARGET_name> [CommandFile]
//
//      <TARGET_name> File name of FRAME without extention
//      [CommandFile]   Optional ASCII file with a run time parameters.
//
// INPUT
//      RGB files of frame (field) and corresponding .SGN files created by
// module PLINE.
// RUN TIME parameters:
//
//fscanf(f," %d %d %d",&MinSlope,&MaxSlope,&SlopeStep,);
//SEE ALSO FILE "PLiNes.ini"
// OUTPUT
//      TARGET_name.stp - all local max lines;
//      TARGET_name.pn2   - lines after cleaning selected;
//      TARGET_name.pln - Strips selected;
//      LINEDET.002       - result collection for analyses,
//                        includes keyboard information for analyse.
```

```
#include <stdlib.h>
#include <conio.h>
#include <stdio.h>
#include <string.h>
#include <graph.h>
#include <float.h>
#include <afx.h>
#include <fstream.h>
#include <time.h>
#include "mylibmd.h"
#include "COMP_FNC.h"
#include "COr_FNC8.h"
#include <ctype.h>
```

```
short MaxStripNumber= 16;
short DistanceThresh=10;
short DetectorWidthPlus=4;
short MinSlope=-20, MaxSlope=20, SlopeStep=2;
```

```
GOOD_DIR_LINE __huge Lines[300];
AVERAGE_VEC __huge AverageForLines[300];
```

```
struct _videoconfig vc;
double GAMMA=1.0,CORR_THRESH=0,Thresh_mdl=0;
short VOITING=0,MAP=0;
char f_name[40]="_",FILE_name[40]="_",FRAME_Name[40]="_";
```

PCT pict\_target, target\_map;

```
void WriteLines(GOOD_DIR_LINE *L,short MaxN, const char *name);
void WritePears(GOOD_DIR_LINE *L,short MaxN, const char *name,short
PMax);
```

```
int __cdecl compare_array_elem ( const void *elem1,const void *elem2 );
int Introduction(int arg, char *a);
```

```
//=====
=====
```

```
ofstream LineC;
```

```
//=====
```

```
int main(int argc,char* argv[])
```

```
{short i,j;
```

```
char *p,mess[128],clean[]="
```

```
if(Introduction(argc,argv[argc-1])) return-1;
```

```
// PROTOCOL OUTPUT
```

```
// ===== GRAPHICS START
```

```
if(GRAPHICS_START(&vc,GRAPHMODE)) GRAPH_OUT(-1);
```

```
//=====
```

```
//===== TARGET PICTURE name and vertexes
```

```
SCR_PNT target_pos(0,0); // CONSTRUCTOR default 0,0
```

```
pict_target=sign_storage_rgb(argv[1],vc);
```

```
sign_present_RGB(pict_target,target_pos);
```

```
//=====
```

```
SCR_PNT StartPnt(0,0),EndPnt(pict_target.s_cols-1,0);
```

```
short Slope,Start,Stop;
```

```
GOOD_DIR_LINE LocalMax[51];
```

```
//DEBUG
```

```
// GRAPH_OUT(0);
```

```
// LOOP over SLOPE
```

```
for(Slope=MinSlope;Slope<=MaxSlope;Slope+=SlopeStep)
```

```
{
```

```
_settextposition( vc.numtextrows-3,0);
```

```
printf("Slope %d",Slope);
```

```
Start=__max(0,-Slope);
```

```
Stop=__min(pict_target.s_rows,pict_target.s_rows-Slope);
```

```
// LINE Calculation
```

```
for (EndPnt.r=(StartPnt.r=Start)+Slope;
```

```
StartPnt.r<Stop;StartPnt.r++,EndPnt.r++)
```

```
{
```

```
Lines[StartPnt.r]=
```

```
GOOD_DIR_LINE::GOOD_DIR_LINE(StartPnt,EndPnt);
```

```

AverageForLines[StartPnt.r]=LineMoments(pict_target,Lines[StartPnt.r],MAP)
;

    }
//      Line Estimation

short StartRow,Q0,QUp,QDown;
    for (StartRow=Start;StartRow<Stop-
DetectorWidthPlus;StartRow++)
        Lines[StartRow].Qual=Quality(AverageForLines+StartRow);
//DEBUG
#ifdef DBG0
        _clearscreen( _GCLEARSCREEN );
        sign_present_RGB(pict_target,target_pos);
    for (StartRow=Start;StartRow<Stop-DetectorWidthPlus;StartRow++)
        { _moveto(0, StartRow);
          _lineto(10,StartRow+Slope+1);
          _moveto(255, StartRow+Slope+1 );

          _lineto(255+(short)(Lines[StartRow].Qual*0.5),StartRow+Slope+1);
          _settextposition( vc.numtextrows-2,0);
          _outtext( clean );
          _settextposition( vc.numtextrows-2,0);
          sprintf(mess,"Quality= %6g ",Lines[StartRow].Qual );
          _outtext( mess);*/
        }
    getch();
#endif
//      Line Selection
    for (QUp=0,StartRow=__max(0,-Slope);
        StartRow<Stop;StartRow++)
        {
            Q0=Lines[StartRow].Qual;
            QDown=(StartRow!=Stop-1)?Lines[StartRow].Qual:0;
            if((Q0>=QDown)&&(Q0>=QUp))
                {LocalMax[50]=Lines[StartRow];
                //including in consideration
                qsort((void*)LocalMax,51,sizeof(GOOD_DIR_LINE
),
                                compare_GOOD_DIR_LINE);

                }
            QUp=Q0;
        }
    }
//      End Slope LOOP
CString ProName(argv[1]);
ProName+=".pln";
WriteLines(LocalMax,51 , (const char *)ProName);

```

```

// line grouping
PT st_main,end_main,st_scnd,end_scnd;
    for(i=0;i<51;i++)
        if(LocalMax[i].Qual>0)
            {st_main=LocalMax[i].Start_p();
            end_main=LocalMax[i].End_p();
            for(j=i+1;j<51;j++)
                if(LocalMax[j].Qual>0)
                    {st_scnd=LocalMax[j].Start_p();
                    end_scnd=LocalMax[j].End_p();
                    if((fabs(st_main.v-st_scnd.v)<DistanceThresh)||
                    (fabs(end_main.v-end_scnd.v)<DistanceThresh))
                        LocalMax[j].Qual=0.0;
                    }
            }
    }
    qsort((void*)LocalMax,51,sizeof(GOOD_DIR_LINE),
        compare_GOOD_DIR_LINE);

    ProName=argv[1];
    ProName+=".stp";
    WriteLines(LocalMax,51 , (const char *)ProName);
    ProName=argv[1];
    ProName+=".pn2";

    WritePears(LocalMax,51 , (const char *)ProName, MaxStripNumber);

    GRAPH_OUT();
    pict_target.free_PCT();
    return(0);
}

//
=====
=====
int get_number_plines() // INITIALISATION GRAPHICMODE, GET SCALE
{
    GRAPH_OUT();
    cout << " MinSlope -20, MaxSlope 20, SlopeStep2 DistanceThresh 10
    MAP \n";
    cout <<
        " NTSC 0 ColorPlan1 1 New_plan 2 RGB 3 LUMIN_THR 4 IHS 5
    \n";

    cin >>MinSlope>> MaxSlope>>SlopeStep>>DistanceThresh>>MAP;

    // ===== GRAPHICS START
    if(GRAPHICS_START(&vc,GRAPHMODE)) GRAPH_OUT(-1);
    //=====
    return 1;
}

```

```

//=====
int get_number_plines_f(FILE *f) // INITIALISATION GRAPHICMODE, GET
SCALE
{
    fscanf(f, "%T^\n]s ");
    fscanf(f, "%d %d %d %d
%d",&MinSlope,&MaxSlope,&SlopeStep,&DistanceThresh,
                                     &MAP);

    // Threshold for histogramme      Number Of Points
    // for CorrThresh
    return 1;
}
//=====INTRODUCTION
int Introduction(int arg, char *a)
{
    int FLG_F=0;
    FILE *datainf;
    short n=0;
    if((arg != 2) && (arg !=3))
    {
        printf(" target-file \n");
        FLG_F=0;
        return(1);
    }
    else
        if(arg ==3)
        {
            FLG_F=1;
            if(!(datainf=fopen(a,"r"))) return 0;
        }
    if(FLG_F)
        {
            get_number_plines_f(datainf); //      GET SCALE AND PARAMETERS
            fclose (datainf);
        }
    else
        get_number_plines();
    return 0;
}
//=====
=====
void WriteLines(GOOD_DIR_LINE *L,short MaxN, const char *name)
{
    LineC.open(name,ios::out || ios::trunc);
    if(LineC.fail())
        {
            LineC.clear(0);
            cout << "CAN NOT OPEN StripCollection";
            GRAPH_OUT(-1);
        }
    LineC<<" # \t st_X \t st_Y \t end_X \t end_y \t Value\n";
    // OUTPUT all 51 line

```



```

short i;
PT st,end;
    for(i=0;i<MaxN;i++)
        if((L+i)->Qual>0)
            {st=(L+i)->Start_p();end=(L+i)->End_p();
              LineC<< i<<" \t"<< (short)st.u<<" \t"<<(short)st.v+2<<
                " \t"<< end.u<<" \t"<< end.v+2<<
                " \t"<<(L+i)->Qual<<"\n";
            }
LineC.close();
}
//=====
==
void WritePears(GOOD_DIR_LINE *L,short MaxN, const char *name,short
PMax)
{
    LineC.open(name,ios::out||ios::trunc);
    if(LineC.fail())
        {LineC.clear(0);
          cout << "CAN NOT OPEN StripCollection";
          GRAPH_OUT(-1);
        }
    LineC<<" Strip_Collection \n";
    // OUTPUT 16 pears
    short i,n=0,j;
    PT st,end,st2,end2;
        for(i=0;i<MaxN-1;i++)
            if((L+i)->Qual>0)
                {st=(L+i)->Start_p();end=(L+i)->End_p();
                  for(j=i+1;(n<PMax) && (j<MaxN);j++)
                      if((L+j)->Qual>0)
                          {n++;
                            LineC<< (short)st.u<<" \t"<<
                              (short)st.v+2<<" \t"<< end.u<<" \t"<< end.v+2<<"\n";
                            st2=(L+j)->Start_p();end2=(L+j)->End_p();
                            LineC<< (short)st2.u<<" \t"<<
                              (short)st2.v+2<<" \t"<< end2.u<<" \t"<<
                              end2.v+2<<"\n";
                          }
                }
    LineC.close();
}
//=====
==

```

```

ORIGIN = PWB
ORIGIN_VER = 2.0
PROJ = LN_DEC2
PROJFILE = LN_DEC2.mak
BUILDDIR = obj
DEBUG = 1

```

```

BRFLAGS = /o obj\$(PROJ).bsc
BSCMAKE = bscmake
SBRPACK = sbrpack
NMAKEBSC1 = set
NMAKEBSC2 = nmake
BROWSE = 1
CC = cl
CFLAGS_G = /W2 /BATCH /FR$*.sbr /Zn
CFLAGS_D = /f /Zi /Od
CFLAGS_R = /f- /Ot /Oi /Ol /Oe /Og /Gs
CXX = cl
CXXFLAGS_G = /AL /W4 /G2 /D_DOS /BATCH /FR$*.sbr /Zn
CXXFLAGS_D = /f- /Od /FPI87 /Zi /DMICI /DSINGLE_WIN /D_DEBUG
CXXFLAGS_R = /f- /Ot /Ol /Og /Oe /Oi /FPI87 /Gs /DMICI /DSINGLE_WIN
MAPFILE_D = NUL
MAPFILE_R = NUL
LFLAGS_G = /NOI /STACK:32000 /BATCH /ONERROR:NOEXE
LFLAGS_D = /CO /FAR /PACKC
LFLAGS_R = /EXE /FAR /PACKC
LINKER = link
ILINK = ilink
LRF = echo > NUL
ILFLAGS = /a /e
LLIBS_R = LAFXCR
LLIBS_D = LAFXCRD
LLIBS_G = graphics
CVFLAGS = /25 /S
RUNFLAGS = \ilya\pnewline\tst\0234e ln_dec.ini

```

```

FILES = ..\LIB\VICALLOC.CPP ..\LIB\PROJECTN7.CPP ..\LIB\PIC_M7.CPP\
        ..\LIB\RES_MCH7.CPP ..\LIB\L_TRN7.CPP LN_TPL1.CPP
LN_DEC2.CPP
OBJS = obj\VICALLOC.obj obj\PROJECTN7.obj obj\PIC_M7.obj
        obj\RES_MCH7.obj
        obj\L_TRN7.obj obj\LN_TPL1.obj obj\LN_DEC2.obj
SBRs = obj\VICALLOC.sbr obj\PROJECTN7.sbr obj\PIC_M7.sbr
        obj\RES_MCH7.sbr
        obj\L_TRN7.sbr obj\LN_TPL1.sbr obj\LN_DEC2.sbr

```

```
all: obj\$(PROJ).exe
```

```
.SUFFIXES:
```

.SUFFIXES:

.SUFFIXES: .obj .sbr .cpp

```
obj\VICALLOC.obj : ..\LIB\VICALLOC.CPP C:\C700\INCLUDE\stdlib.h\
C:\C700\INCLUDE\stdio.h C:\C700\INCLUDE\vmemory.h\
C:\C700\INCLUDE\malloc.h
```

!IF \$(DEBUG)

```
@$(CXX) @<<obj\$(PROJ).rsp
```

/c \$(CXXFLAGS\_G)

```
$(CXXFLAGS_D) /Foobj\VICALLOC.obj ..\LIB\VICALLOC.CPP
```

&lt;&lt;

!ELSE

```
@$(CXX) @<<obj\$(PROJ).rsp
```

/c \$(CXXFLAGS\_G)

```
$(CXXFLAGS_R) /Foobj\VICALLOC.obj ..\LIB\VICALLOC.CPP
```

&lt;&lt;

!ENDIF

```
obj\VICALLOC.sbr : ..\LIB\VICALLOC.CPP C:\C700\INCLUDE\stdlib.h\
C:\C700\INCLUDE\stdio.h C:\C700\INCLUDE\vmemory.h\
C:\C700\INCLUDE\malloc.h
```

!IF \$(DEBUG)

```
@$(CXX) @<<obj\$(PROJ).rsp
```

/Zs \$(CXXFLAGS\_G)

```
$(CXXFLAGS_D) /FRobj\VICALLOC.sbr ..\LIB\VICALLOC.CPP
```

&lt;&lt;

!ELSE

```
@$(CXX) @<<obj\$(PROJ).rsp
```

/Zs \$(CXXFLAGS\_G)

```
$(CXXFLAGS_R) /FRobj\VICALLOC.sbr ..\LIB\VICALLOC.CPP
```

&lt;&lt;

!ENDIF

```
obj\PROJECTN7.obj : ..\LIB\PROJECTN7.CPP C:\C700\INCLUDE\graph.h\
C:\C700\INCLUDE\stdlib.h C:\C700\INCLUDE\iostream.h\
..\LIB\projectn7.h\
```

```
C:\C700\INCLUDE\ios.h C:\C700\INCLUDE\streamb.h\
```

```
C:\C700\INCLUDE\istream.h C:\C700\INCLUDE\ostream.h\
```

```
C:\C700\INCLUDE\math.h
```

!IF \$(DEBUG)

```
@$(CXX) @<<obj\$(PROJ).rsp
```

/c \$(CXXFLAGS\_G)

```
$(CXXFLAGS_D) /Foobj\PROJECTN7.obj ..\LIB\PROJECTN7.CPP
```

&lt;&lt;

!ELSE

```
@$(CXX) @<<obj\$(PROJ).rsp
```

/c \$(CXXFLAGS\_G)

```
$(CXXFLAGS_R) /Foobj\PROJECTN7.obj ..\LIB\PROJECTN7.CPP
```

&lt;&lt;

!ENDIF

```
obj\PROJECTN7.sbr : ..\LIB\PROJECTN7.CPP C:\C700\INCLUDE\graph.h
C:\C700\INCLUDE\stdlib.h C:\C700\INCLUDE\iostream.h
..\LIB\projctn7.h
C:\C700\INCLUDE\ios.h C:\C700\INCLUDE\streamb.h
C:\C700\INCLUDE\istream.h C:\C700\INCLUDE\ostream.h
C:\C700\INCLUDE\math.h
```

!IF \$(DEBUG)

```
@$(CXX) @<<obj\$(PROJ).rsp
/Zs $(CXXFLAGS_G)
$(CXXFLAGS_D) /Fobj\PROJECTN7.sbr ..\LIB\PROJECTN7.CPP
<<
```

!ELSE

```
@$(CXX) @<<obj\$(PROJ).rsp
/Zs $(CXXFLAGS_G)
$(CXXFLAGS_R) /Fobj\PROJECTN7.sbr ..\LIB\PROJECTN7.CPP
<<
```

!ENDIF

```
obj\PIC_M7.obj : ..\LIB\PIC_M7.CPP C:\C700\INCLUDE\stdlib.h
C:\C700\INCLUDE\stdio.h C:\C700\INCLUDE\graph.h
C:\C700\INCLUDE\math.h
C:\C700\INCLUDE\io.h C:\C700\INCLUDE\fcntl.h
C:\C700\INCLUDE\string.h
C:\C700\INCLUDE\float.h C:\C700\INCLUDE\malloc.h ..\LIB\phdr.h
..\LIB\vicalloc.h ..\LIB\pic_mch7.h C:\C700\INCLUDE\vmemory.h
..\LIB\projctn7.h
```

!IF \$(DEBUG)

```
@$(CXX) @<<obj\$(PROJ).rsp
/c $(CXXFLAGS_G)
$(CXXFLAGS_D) /Foobj\PIC_M7.obj ..\LIB\PIC_M7.CPP
<<
```

!ELSE

```
@$(CXX) @<<obj\$(PROJ).rsp
/c $(CXXFLAGS_G)
$(CXXFLAGS_R) /Foobj\PIC_M7.obj ..\LIB\PIC_M7.CPP
<<
```

!ENDIF

```
obj\PIC_M7.sbr : ..\LIB\PIC_M7.CPP C:\C700\INCLUDE\stdlib.h
C:\C700\INCLUDE\stdio.h C:\C700\INCLUDE\graph.h
C:\C700\INCLUDE\math.h
C:\C700\INCLUDE\io.h C:\C700\INCLUDE\fcntl.h
C:\C700\INCLUDE\string.h
C:\C700\INCLUDE\float.h C:\C700\INCLUDE\malloc.h ..\LIB\phdr.h
..\LIB\vicalloc.h ..\LIB\pic_mch7.h C:\C700\INCLUDE\vmemory.h
..\LIB\projctn7.h
```

!IF \$(DEBUG)

```

    @$(CXX) @<<obj\$(PROJ).rsp
/Zs $(CXXFLAGS_G)
$(CXXFLAGS_D) /Fobj\PIC_M7.sbr ..\LIB\PIC_M7.CPP
<<
!ELSE
    @$(CXX) @<<obj\$(PROJ).rsp
/Zs $(CXXFLAGS_G)
$(CXXFLAGS_R) /Fobj\PIC_M7.sbr ..\LIB\PIC_M7.CPP
<<
!ENDIF

obj\RES_MCH7.obj : ..\LIB\RES_MCH7.CPP C:\C700\INCLUDE\stdlib.h\
    C:\C700\INCLUDE\vmemory.h ..\LIB\pic_mch7.h ..\LIB\res_mch7.h\
    C:\C700\INCLUDE\graph.h ..\LIB\projctn7.h C:\C700\INCLUDE\math.h
!IF $(DEBUG)
    @$(CXX) @<<obj\$(PROJ).rsp
/c $(CXXFLAGS_G)
$(CXXFLAGS_D) /Foobj\RES_MCH7.obj ..\LIB\RES_MCH7.CPP
<<
!ELSE
    @$(CXX) @<<obj\$(PROJ).rsp
/c $(CXXFLAGS_G)
$(CXXFLAGS_R) /Foobj\RES_MCH7.obj ..\LIB\RES_MCH7.CPP
<<
!ENDIF

obj\RES_MCH7.sbr : ..\LIB\RES_MCH7.CPP C:\C700\INCLUDE\stdlib.h\
    C:\C700\INCLUDE\vmemory.h ..\LIB\pic_mch7.h ..\LIB\res_mch7.h\
    C:\C700\INCLUDE\graph.h ..\LIB\projctn7.h C:\C700\INCLUDE\math.h
!IF $(DEBUG)
    @$(CXX) @<<obj\$(PROJ).rsp
/Zs $(CXXFLAGS_G)
$(CXXFLAGS_D) /Fobj\RES_MCH7.sbr ..\LIB\RES_MCH7.CPP
<<
!ELSE
    @$(CXX) @<<obj\$(PROJ).rsp
/Zs $(CXXFLAGS_G)
$(CXXFLAGS_R) /Fobj\RES_MCH7.sbr ..\LIB\RES_MCH7.CPP
<<
!ENDIF

obj\L_TRN7.obj : ..\LIB\L_TRN7.CPP C:\C700\INCLUDE\stdlib.h\
    C:\C700\INCLUDE\stdio.h C:\C700\INCLUDE\conio.h\
    C:\C700\INCLUDE\malloc.h ..\LIB\wicalloc.h ..\LIB\lin_trn7.h\
    C:\C700\INCLUDE\vmemory.h ..\LIB\projctn7.h ..\LIB\res_mch7.h\
    ..\LIB\pic_mch7.h C:\C700\INCLUDE\math.h
    C:\C700\INCLUDE\graph.h
!IF $(DEBUG)
    @$(CXX) @<<obj\$(PROJ).rsp

```

```

/c $(CXXFLAGS_G)
$(CXXFLAGS_D) /Foobj\LN_TRN7.obj ..\LIB\LN_TRN7.CPP
<<
!ELSE
    @$(CXX) @<<obj\$(PROJ).rsp
/c $(CXXFLAGS_G)
$(CXXFLAGS_R) /Foobj\LN_TRN7.obj ..\LIB\LN_TRN7.CPP
<<
!ENDIF

obj\LN_TRN7.sbr : ..\LIB\LN_TRN7.CPP C:\C700\INCLUDE\stdlib.h\
C:\C700\INCLUDE\stdio.h C:\C700\INCLUDE\conio.h\
C:\C700\INCLUDE\malloc.h ..\LIB\vicalloc.h ..\LIB\lin_trn7.h\
C:\C700\INCLUDE\vmemory.h ..\LIB\projctn7.h ..\LIB\res_mch7.h\
..\LIB\pic_mch7.h C:\C700\INCLUDE\math.h
C:\C700\INCLUDE\graph.h
!IF $(DEBUG)
    @$(CXX) @<<obj\$(PROJ).rsp
/Zs $(CXXFLAGS_G)
$(CXXFLAGS_D) /FRobj\LN_TRN7.sbr ..\LIB\LN_TRN7.CPP
<<
!ELSE
    @$(CXX) @<<obj\$(PROJ).rsp
/Zs $(CXXFLAGS_G)
$(CXXFLAGS_R) /FRobj\LN_TRN7.sbr ..\LIB\LN_TRN7.CPP
<<
!ENDIF

obj\LN_TPL1.obj : LN_TPL1.CPP C:\C700\INCLUDE\stdio.h
C:\C700\INCLUDE\stdlib.h\
C:\C700\INCLUDE\conio.h C:\C700\INCLUDE\malloc.h\
C:\C700\INCLUDE\float.h C:\C700\INCLUDE\graph.h\
C:\C700\INCLUDE\string.h C:\C700\INCLUDE\ctype.h match5.h\
..\LIB\pic_mch7.h ..\LIB\projctn7.h ..\LIB\res_mch7.h\
..\LIB\lin_trn7.h C:\C700\INCLUDE\vmemory.h
C:\C700\INCLUDE\math.h
!IF $(DEBUG)
    @$(CXX) @<<obj\$(PROJ).rsp
/c $(CXXFLAGS_G)
$(CXXFLAGS_D) /Foobj\LN_TPL1.obj LN_TPL1.CPP
<<
!ELSE
    @$(CXX) @<<obj\$(PROJ).rsp
/c $(CXXFLAGS_G)
$(CXXFLAGS_R) /Foobj\LN_TPL1.obj LN_TPL1.CPP
<<
!ENDIF

```

```

obj\LN_TPL1.sbr : LN_TPL1.CPP C:\C700\INCLUDE\stdio.h
C:\C700\INCLUDE\stdlib.h\
    C:\C700\INCLUDE\conio.h C:\C700\INCLUDE\malloc.h\
    C:\C700\INCLUDE\float.h C:\C700\INCLUDE\graph.h\
    C:\C700\INCLUDE\string.h C:\C700\INCLUDE\ctype.h match5.h\
    ..\LIB\pic_mch7.h ..\LIB\projctn7.h ..\LIB\res_mch7.h\
    ..\LIB\lin_tm7.h C:\C700\INCLUDE\vmemory.h
C:\C700\INCLUDE\math.h
!IF $(DEBUG)
    @$(CXX) @<<obj\$(PROJ).rsp
/Zs $(CXXFLAGS_G)
$(CXXFLAGS_D) /Fobj\LN_TPL1.sbr LN_TPL1.CPP
<<
!ELSE
    @$(CXX) @<<obj\$(PROJ).rsp
/Zs $(CXXFLAGS_G)
$(CXXFLAGS_R) /Fobj\LN_TPL1.sbr LN_TPL1.CPP
<<
!ENDIF

```

```

obj\LN_DEC2.obj : LN_DEC2.CPP C:\C700\INCLUDE\stdlib.h
C:\C700\INCLUDE\conio.h\
    C:\C700\INCLUDE\stdio.h C:\C700\INCLUDE\string.h\
    C:\C700\INCLUDE\graph.h C:\C700\INCLUDE\float.h\
    C:\C700\MFC\INCLUDE\afx.h C:\C700\INCLUDE\fstream.h\
    C:\C700\INCLUDE\time.h match5.h ..\LIB\vicalloc.h\
    C:\C700\INCLUDE\ctype.h C:\C700\MFC\INCLUDE\afx.in\
    C:\C700\INCLUDE\iostream.h ..\LIB\projctn7.h ..\LIB\pic_mch7.h\
    ..\LIB\res_mch7.h ..\LIB\lin_tm7.h C:\C700\INCLUDE\vmemory.h\
    C:\C700\INCLUDE\ios.h C:\C700\INCLUDE\streamb.h\
    C:\C700\INCLUDE\istream.h C:\C700\INCLUDE\ostream.h\
    C:\C700\INCLUDE\math.h
!IF $(DEBUG)
    @$(CXX) @<<obj\$(PROJ).rsp
/c $(CXXFLAGS_G)
$(CXXFLAGS_D) /Foobj\LN_DEC2.obj LN_DEC2.CPP
<<
!ELSE
    @$(CXX) @<<obj\$(PROJ).rsp
/c $(CXXFLAGS_G)
$(CXXFLAGS_R) /Foobj\LN_DEC2.obj LN_DEC2.CPP
<<
!ENDIF

```

```

obj\LN_DEC2.sbr : LN_DEC2.CPP C:\C700\INCLUDE\stdlib.h
C:\C700\INCLUDE\conio.h\
    C:\C700\INCLUDE\stdio.h C:\C700\INCLUDE\string.h\
    C:\C700\INCLUDE\graph.h C:\C700\INCLUDE\float.h\
    C:\C700\MFC\INCLUDE\afx.h C:\C700\INCLUDE\fstream.h\

```

```

C:\C700\INCLUDE\time.h match5.h ..\LIB\vicalloc.h\
C:\C700\INCLUDE\ctype.h C:\C700\MFC\INCLUDE\afx.in\
C:\C700\INCLUDE\iostream.h ..\LIB\projctn7.h ..\LIB\pic_mch7.h\
..\LIB\res_mch7.h ..\LIB\lin_tm7.h C:\C700\INCLUDE\vmemory.h\
C:\C700\INCLUDE\ios.h C:\C700\INCLUDE\streamb.h\
C:\C700\INCLUDE\istream.h C:\C700\INCLUDE\ostream.h\
C:\C700\INCLUDE\math.h
!IF $(DEBUG)
    @$(CXX) @<<obj\$(PROJ).rsp
/Zs $(CXXFLAGS_G)
$(CXXFLAGS_D) /FRobj\LN_DEC2.sbr LN_DEC2.CPP
<<
!ELSE
    @$(CXX) @<<obj\$(PROJ).rsp
/Zs $(CXXFLAGS_G)
$(CXXFLAGS_R) /FRobj\LN_DEC2.sbr LN_DEC2.CPP
<<
!ENDIF

```

```

obj\$(PROJ).bsc : $(SBRs)
    $(BSCMAKE) @<<
$(BRFLAGS) $(SBRs)
<<

```

```

obj\$(PROJ).exe : $(OBJS)
    -$(NMAKEBSC1) MAKEFLAGS=
    -$(NMAKEBSC2) $(NMFLAGS) -f $(PROJFILE) obj\$(PROJ).bsc

```

```

!IF $(DEBUG)
    $(LRF) @<<obj\$(PROJ).lrf
$(RT_OBJS: = +^
) $(OBJS: = +^
)
$@
$(MAPFILE_D)
$(LIBS: = +^
) +
$(LLIBS_G: = +^
) +
$(LLIBS_D: = +^
)
$(DEF_FILE) $(LFLAGS_G) $(LFLAGS_D);
<<
!ELSE
    $(LRF) @<<obj\$(PROJ).lrf
$(RT_OBJS: = +^
) $(OBJS: = +^
)
$@

```



```

$(MAPFILE_R)
$(LIBS: = +^
) +
$(LLIBS_G: = +^
) +
$(LLIBS_R: = +^
)
$(DEF_FILE) $(LFLAGS_G) $(LFLAGS_R);
<<
!ENDIF
$(LINKER) @obj\$(PROJ).lrf

```

```

.cpp.obj :
!IF $(DEBUG)
    @$(CXX) @<<obj\$(PROJ).rsp
/c $(CXXFLAGS_G)
$(CXXFLAGS_D) /Fo$@ $<
<<
!ELSE
    @$(CXX) @<<obj\$(PROJ).rsp
/c $(CXXFLAGS_G)
$(CXXFLAGS_R) /Fo$@ $<
<<
!ENDIF

```

```

.cpp.sbr :
!IF $(DEBUG)
    @$(CXX) @<<obj\$(PROJ).rsp
/Zs $(CXXFLAGS_G)
$(CXXFLAGS_D) /FR$@ $<
<<
!ELSE
    @$(CXX) @<<obj\$(PROJ).rsp
/Zs $(CXXFLAGS_G)
$(CXXFLAGS_R) /FR$@ $<
<<
!ENDIF

```

```

run: obj\$(PROJ).exe
    obj\$(PROJ).exe $(RUNFLAGS)

```

```

debug: obj\$(PROJ).exe
    CV $(CVFLAGS) obj\$(PROJ).exe $(RUNFLAGS)

```

```
#ifndef MATCH
#define MATCH
#include "projctn7.h"
#include "pic_mch7.h"
#include "res_mch7.h"
#include "lin_tm7.h"
// #include "tem_plt7.h"

#define NAME_LENGTH 40
#define GRAPHMODE _VRES256COLOR

// COLOR_RGB INTER_pix_color_rgb(PCT p1, PT PT_now);
#endif
```

```

//      STRIP SELECTION
//      Module gets a strips after line detection (module plines
// and selects two lines with a maximumm informational
// contents (frequency approach).
// COMMAND STRING
//
// Ins_corr <TARGET_name> [CommandFile]
//
// <TARGET_name> File name of FRAME without extention
// $ [CommandFile] Optional ASCI file with a run time parameters.
//
// INPUT
//=====ATTENTION
// LN_DEC2 used .pn2 files - output PLNEW modules
// RGB files of frame (field) and corresponding .SGN files created by
// module PLINE.
// RUN TIME parameters:
//
// <four number> -shift of strips vertexes relative to original>
// <V H_low H_up Hight Width> -Five integer:
//      V          - The highest vertical harmonic;
//      H_low, H_up - The lowest and highest horisontal harmonics;
//      Hight, Width - Hight and Width of windowfor analyses.
// <Gap HalfScrWeight PosWeight> - Integer and two float <1:
//      Gap          - range unsensetivites to screen position;
//      HalfScrWeight - Additional weght if an upper line is
//                      in upper half of screen
//      PosWeight    - Additional weght if miuddle line belowe
//                      then middle line of another line.
//SEE ALSO FILE "LN_DEC.Ini"
// OUTPUT
//      TARGET_name.SG2      - Strips selected;
//      LINEDET.002          - result collection for analyses,
//                      includes keyboard information to analyse.
//=====ATTENTION
// LN_DEC2 OUTPUT
//      TARGET_name.SG2      - Strips selected;
//      LINEDET.002          - result collection for analyses,
//                      includes keyboard information to analyse.

#include <stdlib.h>
#include <conio.h>
#include <stdio.h>
#include <string.h>
#include <graph.h>
#include <float.H>
#include <afx.h>
#include <fstream.h>
#include <time.h>

```

```

#include "match5.h"
#include "vicalloc.h"
#include <ctype.h>

short PRESENT_HIGHT;
short GAP=8;
double HalfScrWeight=0.1, PosWeight=0.1;

short TAG_hight;
struct _videoconfig vc;
SCR_PNT vrt_target[4];
FILE *datres;
double sh[4]={0,0,0,0};
double GAMMA=1.0,CORR_THRESH=0,Thresh_mdl=0;
short VOITING=0,MAP=0;
char f_name[40]="_",FILE_name[40]="_",FRAME_Name[40]="_",
    STRING_name[40]="_",SIGN_name[40]="_";
PCT pict_target, target_map;
short win_hight,win_lenth;
short HOR_HARM_l,HOR_HARM_h,VERT_HARM_1;

int __cdecl compare_array_elem ( const void *elem1,const void *elem2 );
int PictureInf(const char *name,SCR_PNT *vertexes,short n);

//int picture_inf_num_new(char *name,SCR_PNT *vertexes,short n);
void get_shift_f(FILE *f,double * sh); // INITIALISATION GRAPHICMODE,
GET SCALE
void get_shift(double * sh); // INITIALISATION GRAPHICMODE, GET SCALE
int get_number_3();
int get_number_3_f(FILE *f);
double GlobalWinCalc(PCT target_map,
    SCR_PNT win_size, short winpos);

//=====
=====
ofstream LineC;
//===== OpenStripCollection
void OpenStripCol(const char*name)
{ LineC.open(name,ios::out|ios::app|ios::out|ios::nocreate);
  if(LineC.fail())
  {LineC.clear(0);
   LineC.open(name,ios::out|ios::app|ios::out|ios::noreplace);
   if(LineC.fail())
   {LineC.clear(0);
    cout << "CAN NOT OPEN StripCollection";
    GRAPH_OUT(-1);
   }
  }
  else
  LineC<<"StripAnalysCollection \n";
}

```

```

LineC<<"Name #1\VERT_HARM_1 \t"<<
" HOR_HARM_1 \t HOR_HARM_h \t WindowLength\n";
}
//=====
int main(int argc,char* argv[])
{
char *p,mess[128],clean[]="";
SCR_PNT line_vrt[20][4];
int FLG_F=0;
FILE *datainf;
short n=0;
SCR_PNT t_pos;
if((argc != 2) && (argc !=3))

{
printf(" target-file \n");
FLG_F=0;
return(1);
}
else
if(argc ==3)
{FLG_F=1;
if(!(datainf=fopen(argv[2],"r"))) return 0;
}
if(FLG_F)
{get_shift_f(datainf,sh); // GET SCALE
get_number_3_f(datainf);
}
else
{get_shift(sh); // GET SCALE
get_number_3();
}
// PROTOCOL OUTPUT
CString ProName("linedet.002");
strcpy(f_name,argv[1]);
OpenStripCol((const char *)ProName);
LineC<<f_name<<" \t" <<VERT_HARM_1<<" \t"<< HOR_HARM_1<<" \t"
<<HOR_HARM_h<<" \t"<<win_lenth<<"\n";
LineC<<"GAP="<< GAP<<" \tHalfScrWeight="<<HalfScrWeight<<
" \tPosWeight="<<PosWeight<<"\n";
LineC<<" Line # \t Hight \t Value \t Comm \n";
// ===== GRAPHICS START
if(GRAPHICS_START(&vc,GRAPHMODE)) GRAPH_OUT(-1);
//=====
//===== TARGET PICTURE name and vertexes
SCR_PNT target_pos; // CONSTRUCTOR default 0,0
pict_target=sign_storage_rgb(argv[1],vc);
//===== PROTOTYPE initialisation
short j,k;

```

```

double res[20];
for(j=0;j<20;res[j++]=0.0);
CString FName(' ',80);
    FName=argv[1];
    FName+="pn2";
while(PictureInf((const char*)FName,vrt_target, n++))
{
    SCR_PNT vrt[4];
    match_vertex( vrt_target);
    for(j=0;j<4;j++)
    {
        vrt[j].c=vrt_target[j].c;
        k=((j%2)?j-1:j+1);
        if (fabs(sh[j])<1)
        {vrt[j].r=(sh[j]*(short)((vrt_target[j].r-vrt_target[k].r)*sh[j])+
          vrt_target[j].r.vrt_target[j].r);
        }
        else
            vrt[j].r=vrt_target[j].r+sh[j];
    }
#define CALC_HIGHT 16
    TAG_hight=(short)(vrt[2].r-vrt[3].r+vrt[1].r-vrt[0].r+2)*0.5;
    if(TAG_hight<10) continue;
    target_map=linear_transform_cont(pict_target, vrt, CALC_HIGHT );
    if(!target_map.s_cols)
        {printf("TOO NARROW LINES");continue;}
    match_vertex( vrt);

    PRESENT_HIGHT=16;
    _clearscreen(_GCLEARSCREEN);
    target_pos.c=10;
    target_pos.r=10;
    sign_present_RGB( target_map,target_pos);
    //=====
    SCR_PNT win_size(128,4);
    short winpos;

    win_size.r=(win_hight>0)?__min(win_hight,CALC_HIGHT):CALC_HIGHT;
    win_size.c=(win_lenth>0)?__min(win_lenth,target_map.s_cols)
        :target_map.s_cols;
    winpos=(CALC_HIGHT-win_size.r)/2;

    const double scale=(double)PRESENT_HIGHT/CALC_HIGHT;
    // RESULTS AND VERTEXES
        res[n-1]= GlobalWinCalc(target_map,win_size, winpos);
    for (j=0;j<4;j++)
        line_vrt[n-1][j]=vrt[j];
    target_map.free_PCT();
    //=====

```

```

LineC<< n-1<<" \t"<< TAG_hight<<" \t"<< res[n-1]<<" \t";
//LineC<<" Line # \t Hight \t Value \t Comm_1 \t COMM_2 \n";
    sprintf(mess,"File Name ");
    _settextposition( vc.numtextrows-2,0);
    _outtext( clean );
    _settextposition( vc.numtextrows-2,0);
    _outtext( mess );
    p=mess;
    while(!isspace( (int)( *p++=(char)getche() ));
        *(-p)='\0';
    LineC<<mess<<"\t\n";
//=====
}
// ——— DECISION MAKING
short i,nl[5], up_ln[5],top_ln[5];
short best=0,scnd=0;
double      *nmb[20],weight[5]={0,0,0,0,0};

    for(i=0;i<20;i++) nmb[i]=res+i;
    qsort((void*)nmb,20,sizeof(double*),compare_array_elem );
    for(i=0;i<5;i++)
        {if(!(*nmb[i])) break;
        nl[i]=nmb[i]-res;
        up_ln[i]=line_vrt[nl[i]][3].r+line_vrt[nl[i]][0].r;
        top_ln[i]=line_vrt[nl[i]][2].r+line_vrt[nl[i]][2].r;
        weight[i]=*nmb[i];
        }
    if(weight[0])
        {if((up_ln[1]+top_ln[1])>(up_ln[0]+top_ln[0]-GAP))
            weight[1]*=(1.0+PosWeight);
        if(up_ln[1]>pict_target.s_rows)
            weight[1]*=(1.0+HalfScrWeight);
        if(up_ln[0]>pict_target.s_rows)
            weight[0]*=(1.0+HalfScrWeight);
        scnd=(best=(weight[0]<weight[1])?1:0)?0:1;

//——— DECISION OUTPUT
        target_pos.c=10;
        target_pos.r=10;
        _clearscreen(_ GCLEARSCREEN);
        target_map=linear_transform_cont(pict_target, line_vrt[nl[best]],16 );
        sign_present_RGB( target_map,target_pos);
        target_map.free_PCT();

        if(weight[scnd])
        {
            target_pos.r+=20;
            target_map=linear_transform_cont(pict_target, line_vrt[nl[scnd]],16 );
            sign_present_RGB( target_map,target_pos);
        }
    }

```

```

        target_map.free_PCT();
    }
    sprintf(mess,"Result 1_res= %6g w= %6g 2_res=%6f W=%6f \0",
        *nmb[best],weight[best],*nmb[scnd],weight[scnd]);

char Sn1[20];
    _settextposition( vc.numtextrows-2,1);
    _outtext( clean );
    _settextposition( vc.numtextrows-2,0);
    _outtext( mess );
    p=Sn1;
    while(!isspace( (int)( *p++=(char)getche() ) ));
    *(-p)='\0';
    strcat(mess,Sn1);
LineC<< mess<<'\n';
LineC.close();
//picture presentation
    _clearscreen(_ GCLEARSCREEN);
    target_pos.c=0;
    target_pos.r=0;
    target_map=linear_transform_cont(pict_target, line_vrt[nl[best]],16 );
    sign_present_RGB( pict_target,target_pos);
    _setcolor( color_num(240,240,240));
    _moveto(line_vrt[nl[best]][3].c,line_vrt[nl[best]][3].r);
    _lineto(line_vrt[nl[best]][0].c,line_vrt[nl[best]][0].r);
    _moveto(line_vrt[nl[best]][2].c,line_vrt[nl[best]][2].r);
    _lineto(line_vrt[nl[best]][1].c,line_vrt[nl[best]][1].r);
    getch();
    if(weight[scnd])
    {
        _setcolor( color_num(240,240,0));
        _moveto(line_vrt[nl[scnd]][3].c,line_vrt[nl[scnd]][3].r);
        _lineto(line_vrt[nl[scnd]][0].c,line_vrt[nl[scnd]][0].r);
        _moveto(line_vrt[nl[scnd]][2].c,line_vrt[nl[scnd]][2].r);
        _lineto(line_vrt[nl[scnd]][1].c,line_vrt[nl[scnd]][1].r);
        getch();
    }

    strcat(strcpy(f_name,argv[1]),".sg2");
    LineC.open(f_name,ios::out);
    LineC<<"After Str Selection\n";
    LineC<<argv[1]<<" ____ "<<Sn1<<" \t" <<Sn1<<'\n';
    for(j=0;j<4;j++)
        LineC<<line_vrt[nl[best]][j].c<<" \t"<<line_vrt[nl[best]][j].r<<'\n';
    if(weight[scnd])
        for(j=0;j<4;j++)
            LineC<<line_vrt[nl[scnd]][j].c<<" \t"<<line_vrt[nl[scnd]][j].r<<'\n';
    LineC.close();
//END DECISION

```



```

    }
else
{
    LineC.open(f_name,ios::out);
    LineC<<"After Str Selection\n";
    LineC<<argv[1]<<" ____ "<<"____\t____\n";
    LineC.close();

    cout <<" Lines not proposed";
}
GRAPH_OUT();
pict_target.free_PCT();
target_map.free_PCT();
fclose(datres);
return(0);
}

//=====
=====
int __cdecl compare_array_elem ( const void *elem1,const void *elem2 )
{int i;
 double a;
 a=*(double **)elem1-*(double **)elem2;
 i=(a?((a<0)?1:-1):0);
 return i;
}
//
=====
=====
void get_shift_f(FILE *f,double * sh) // INITIALISATION GRAPHICMODE,
GET SCALE
{
    fscanf(f," %lf %lf %lf %lf",sh,sh+1,sh+2,sh+3);
}
//
=====
=====
void get_shift(double * sh) // INITIALISATION GRAPHICMODE, GET SCALE
{int i;
    cout<<"vertexes shift over rows ( top_right, bottom_right, bottom_left,
top_left \n";
    for (i=0;i<4; i++)
        cin>>sh[i];
}
//=====
=====
int get_number_3()
{
    GRAPH_OUT();

```

```

cout << " VERT_HARM<16 HOR_HARM_start<16
HOR_HARM_stop,win_hight win_lenth";
cin
>>VERT_HARM_1>>HOR_HARM_I>>HOR_HARM_h>>win_hight>>win_lent
h;
cout<<" GAP HalfScrWeight % PosWeight % ";
cin>>GAP>>HalfScrWeight>>PosWeight;
HalfScrWeight/=100.0;
PosWeight/=100.0;
// ===== GRAPHICS START
if(GRAPHICS_START(&vc,GRAPHMODE)) GRAPH_OUT(-1);
//=====
return 0;
}
//=====
int get_number_3_f(FILE *f)
{ fscanf(f," %d %d %d %d %d",&VERT_HARM_1,
&HOR_HARM_I,&HOR_HARM_h,&win_hight,&win_lenth);
fscanf(f," %d %lf %lf",&GAP,&HalfScrWeight,&PosWeight);
HalfScrWeight/=100.0;
PosWeight/=100.0;
return 1;
}
//=====
int PictureInf(const char *name,SCR_PNT *vertexes,short n)
{int i,j;
ifstream datfp;
char new_str[80];
int r,FLG=0;
datfp.open(name,ios::in);
if(datfp.fail())
{datfp.clear(0);
cout << "CAN NOT OPEN InfFile "<< name<<"\n";
GRAPH_OUT(-1);
}
datfp.getline(new_str,80);
for(j=0;j<n+1;j++)
for(i=0;i<4;i++)
{datfp>>(vertexes+i)->c>>(vertexes+i)->r;
if(datfp.eof())
{datfp.close (); return 0;}
}
datfp.close ();
return 1;
}
//=====

```

```

#include <stdio.h>
#include <stdlib.h>
#include <conio.h>
#include <malloc.h>
#include <float.h>
#include <graph.h>
#include <string.h>
#include <ctype.h>

#include "match5.h"
#include "tem_plt7.h"
#include "pic_mch7.h"
#define MAX_LINE 1024

extern short TAG_hight;
extern struct _videoconfig vc;
extern char f_name[40], FILE_name[40], FRAME_Name[40],
          STRING_name[40], SIGN_name[40];
typedef struct
    {double md[3],var[3];} AVERAGE_VEC;
//=====
===
extern short HOR_HARM_I;
extern short HOR_HARM_h;
extern short VERT_HARM_1 ;
//-----
double      calcul_power(PCT win,short x_scale, double *vr)
{const short
VERT_HARM=VERT_HARM_1*2,HOR_HARM=HOR_HARM_h*2,
    HOR_HARM_s=HOR_HARM_I?HOR_HARM_I*2-1:HOR_HARM_I;
const short THR=16;
double sum[19][19];
COLOR_RGB col;
double pow=0,mid=0;
short c1;
long l_now;
long n;
short h_t,v_t,x,y,h,v,half_x,half_y,quot_x,quot_y;
    n=win.s_cols*win.s_rows;
    half_y=win.s_rows>>1;
    quot_y=win.s_rows>>2;
//!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
    half_x=x_scale>>1;
    quot_x=x_scale>>2;
//    half_x=win.s_rows>>1;
//    quot_x=win.s_rows>>2;
//    half_x=win.s_cols>>1;
//    quot_x=win.s_cols>>2;
//!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!

```

```

    for(h=0;h<HOR_HARM+1;h++)
    for(v=0;v<VERT_HARM+1;v++)
        sum[v][h]=0;
    for(y=0;y<win.s_rows;y++)
    for(x=0;x<win.s_cols;x++)
    {
        col=win.get_pixel(y,x);
        l_now=col.r+col.b+col.g;
        l_now=(l_now>THR)?l_now:0;
        mid+=l_now;
        pow+=l_now*l_now;
        for(v=0;(v<VERT_HARM+1) &&(((v+1)>>1)<=quot_y);v++)
        {
            v_t=y*((v+1)>>1); // HARMONIC #=(v+1)>>1
            v_t=(v_t+(v & 0x0001 ? quot_y:0))/half_y;
            v_t &= 0x0001; //1 if y_pos in 2nd half
            c1= v_t?l_now:-l_now;
        }
    }

    for(h=HOR_HARM_s;(h<HOR_HARM+1)&&(((h+1)>>1)<=quot_x);h++)
    {
        h_t=x*((h+1)>>1);
        h_t=(h_t+(h & 0x0001 ? quot_x:0))/half_x;
        h_t &= 0x0001;
        sum[v][h]+= h_t*c1:-c1;
    }
}

double s0,dd;
for(s0=h=0;h<HOR_HARM+1;h++)
for(v=0;v<VERT_HARM+1;v++)
    if(h||v)
        s0+=(double) sum[v][h]*sum[v][h];

s0/=n;
*vr=(dd=(pow-mid*mid/n)?s0/dd:0;
//return add_out;
return (s0/(n*100.0));
}
//=====
double GlobalWinCalc(PCT target_map,
SCR_PNT win_size, short winpos)
{
    double centr_pow;
    double pow_now;
    long n_fr=0;
    double mid_pow=0;
    double rev_fr;
    PCT win(win_size.c,win_size.r);
    SCR_PNT st_t,st_win;

```

```
st_win.r=winpos;
for (st_win.c=0;st_win.c<target_map.s_cols-win_size.c+1;st_win.c++)
{
    win.load_template(target_map,st_win);
    pow_now=calcul_power(win,target_map.s_rows,&centr_pow);
    mid_pow+=pow_now;
    n_fr++;
}
rev_fr=1.0/n_fr;
mid_pow*=rev_fr;

win.free_PCT();
return mid_pow;
}
//=====
=====
```

```

ORIGIN = PWB
ORIGIN_VER = 2.0
PROJ = LNS_CORR
PROJFILE = LNS_CORR.MAK
BUILDDIR = obj
DEBUG = 0

```

```

BRFLAGS = /o obj\$(PROJ).bsc
BSCMAKE = bscmake
SBRPACK = sbrpack
NMAKEBSC1 = set
NMAKEBSC2 = nmake
BROWSE = 1
CC = cl
CFLAGS_G = /W2 /BATCH /FR$*.sbr /Zn
CFLAGS_D = /f /Zi /Od
CFLAGS_R = /f- /Ot /Oi /Ol /Oe /Og /Gs
CXX = cl
CXXFLAGS_G = /AL /W4 /G2 /D_DOS /BATCH /FR$*.sbr /Zn
CXXFLAGS_D = /f- /Od /FPi87 /Zi /DMICI /DSINGLE_WIN
CXXFLAGS_R = /f- /Ot /Ol /Og /Oe /Oi /FPi87 /Gs /DMICI /DSINGLE_WIN
MAPFILE_D = NUL
MAPFILE_R = NUL
LFLAGS_G = /NOI /STACK:32000 /BATCH /ONERROR:NOEXE
LFLAGS_D = /CO /FAR /PACKC
LFLAGS_R = /EXE /FAR /PACKC
LINKER = link
ILINK = ilink
LRF = echo > NUL
ILFLAGS = /a /e
LLIBS_G = graphics lafxc
CVFLAGS = /25 /S
RUNFLAGS = ..\win4\1S160_0 ..\win4\1S160_ auto1

```

```

FILES = LNS_CORR.CPP ..\LIB\VICALLOC.CPP ..\LIB\PROJCTN7.CPP\
        ..\LIB\PIC_M7.CPP ..\LIB\RES_MCH7.CPP COR_FNC.CPP
COR_WIN.CPP
OBJS = obj\LNS_CORR.obj obj\VICALLOC.obj obj\PROJCTN7.obj
obj\PIC_M7.obj
        obj\RES_MCH7.obj obj\COR_FNC.obj obj\COR_WIN.obj
SBRs = obj\LNS_CORR.sbr obj\VICALLOC.sbr obj\PROJCTN7.sbr
obj\PIC_M7.sbr\
        obj\RES_MCH7.sbr obj\COR_FNC.sbr obj\COR_WIN.sbr

```

```
all: obj\$(PROJ).exe
```

```

.SUFFIXES:
.SUFFIXES:
.SUFFIXES: .obj .sbr .cpp

```

```

obj\LNS_CORR.obj : LNS_CORR.CPP C:\C700\INCLUDE\stdlib.h\
C:\C700\INCLUDE\conio.h C:\C700\INCLUDE\stdio.h\
C:\C700\INCLUDE\string.h C:\C700\INCLUDE\graph.h\
C:\C700\INCLUDE\float.h C:\C700\INCLUDE\io.h
C:\C700\INCLUDE\time.h\
C:\C700\INCLUDE\ctype.h C:\C700\INCLUDE\iostream.h\
C:\C700\MFC\INCLUDE\afx.h LNS_CORR.h cor_win.h
c:\ilya\lib\vicalloc.h\
C:\C700\INCLUDE\ios.h C:\C700\INCLUDE\streamb.h\
C:\C700\INCLUDE\istream.h C:\C700\INCLUDE\ostream.h\
C:\C700\MFC\INCLUDE\afx.inl ..\LIB\projctn7.h ..\LIB\pic_mch7.h\
..\LIB\res_mch7.h c:\ilya\lib\lin_tm7.h C:\C700\INCLUDE\vmemory.h\
C:\C700\INCLUDE\math.h
!IF $(DEBUG)
    @$(CXX) @<<obj$(PROJ).rsp
/c $(CXXFLAGS_G)
$(CXXFLAGS_D) /Foobj\LNS_CORR.obj LNS_CORR.CPP
<<
!ELSE
    @$(CXX) @<<obj$(PROJ).rsp
/c $(CXXFLAGS_G)
$(CXXFLAGS_R) /Foobj\LNS_CORR.obj LNS_CORR.CPP
<<
!ENDIF

obj\LNS_CORR.sbr : LNS_CORR.CPP C:\C700\INCLUDE\stdlib.h\
C:\C700\INCLUDE\conio.h C:\C700\INCLUDE\stdio.h\
C:\C700\INCLUDE\string.h C:\C700\INCLUDE\graph.h\
C:\C700\INCLUDE\float.h C:\C700\INCLUDE\io.h
C:\C700\INCLUDE\time.h\
C:\C700\INCLUDE\ctype.h C:\C700\INCLUDE\iostream.h\
C:\C700\MFC\INCLUDE\afx.h LNS_CORR.h cor_win.h
c:\ilya\lib\vicalloc.h\
C:\C700\INCLUDE\ios.h C:\C700\INCLUDE\streamb.h\
C:\C700\INCLUDE\istream.h C:\C700\INCLUDE\ostream.h\
C:\C700\MFC\INCLUDE\afx.inl ..\LIB\projctn7.h ..\LIB\pic_mch7.h\
..\LIB\res_mch7.h c:\ilya\lib\lin_tm7.h C:\C700\INCLUDE\vmemory.h\
C:\C700\INCLUDE\math.h
!IF $(DEBUG)
    @$(CXX) @<<obj$(PROJ).rsp
/Zs $(CXXFLAGS_G)
$(CXXFLAGS_D) /FRobj\LNS_CORR.sbr LNS_CORR.CPP
<<
!ELSE
    @$(CXX) @<<obj$(PROJ).rsp
/Zs $(CXXFLAGS_G)
$(CXXFLAGS_R) /FRobj\LNS_CORR.sbr LNS_CORR.CPP
<<

```

!ENDIF

obj\VICALLOC.obj : ..\LIB\VICALLOC.CPP C:\C700\INCLUDE\stdlib.h\  
 C:\C700\INCLUDE\stdio.h C:\C700\INCLUDE\vmemory.h\  
 C:\C700\INCLUDE\malloc.h

!IF \$(DEBUG)

@\$(CXX) @<<obj\\$(PROJ).rsp

/c \$(CXXFLAGS\_G)

\$(CXXFLAGS\_D) /Foobj\VICALLOC.obj ..\LIB\VICALLOC.CPP

<<

!ELSE

@\$(CXX) @<<obj\\$(PROJ).rsp

/c \$(CXXFLAGS\_G)

\$(CXXFLAGS\_R) /Foobj\VICALLOC.obj ..\LIB\VICALLOC.CPP

<<

!ENDIF

obj\VICALLOC.sbr : ..\LIB\VICALLOC.CPP C:\C700\INCLUDE\stdlib.h\  
 C:\C700\INCLUDE\stdio.h C:\C700\INCLUDE\vmemory.h\  
 C:\C700\INCLUDE\malloc.h

!IF \$(DEBUG)

@\$(CXX) @<<obj\\$(PROJ).rsp

/Zs \$(CXXFLAGS\_G)

\$(CXXFLAGS\_D) /FRobj\VICALLOC.sbr ..\LIB\VICALLOC.CPP

<<

!ELSE

@\$(CXX) @<<obj\\$(PROJ).rsp

/Zs \$(CXXFLAGS\_G)

\$(CXXFLAGS\_R) /FRobj\VICALLOC.sbr ..\LIB\VICALLOC.CPP

<<

!ENDIF

obj\PROJECTN7.obj : ..\LIB\PROJECTN7.CPP C:\C700\INCLUDE\graph.h\  
 C:\C700\INCLUDE\stdlib.h C:\C700\INCLUDE\iostream.h  
 ..\LIB\projctn7.h\  
 C:\C700\INCLUDE\ios.h C:\C700\INCLUDE\streamb.h\  
 C:\C700\INCLUDE\istream.h C:\C700\INCLUDE\ostream.h\  
 C:\C700\INCLUDE\math.h

!IF \$(DEBUG)

@\$(CXX) @<<obj\\$(PROJ).rsp

/c \$(CXXFLAGS\_G)

\$(CXXFLAGS\_D) /Foobj\PROJECTN7.obj ..\LIB\PROJECTN7.CPP

<<

!ELSE

@\$(CXX) @<<obj\\$(PROJ).rsp

/c \$(CXXFLAGS\_G)

\$(CXXFLAGS\_R) /Foobj\PROJECTN7.obj ..\LIB\PROJECTN7.CPP

<<

!ENDIF



```

obj\PROJECTN7.sbr : ..\LIB\PROJECTN7.CPP C:\C700\INCLUDE\graph.h\
C:\C700\INCLUDE\stdlib.h C:\C700\INCLUDE\iostream.h
..\LIB\projctn7.h\
C:\C700\INCLUDE\ios.h C:\C700\INCLUDE\streamb.h\
C:\C700\INCLUDE\istream.h C:\C700\INCLUDE\ostream.h\
C:\C700\INCLUDE\math.h
!IF $(DEBUG)
    @$(CXX) @<<obj\$(PROJ).rsp
/Zs $(CXXFLAGS_G)
$(CXXFLAGS_D) /FRobj\PROJECTN7.sbr ..\LIB\PROJECTN7.CPP
<<
!ELSE
    @$(CXX) @<<obj\$(PROJ).rsp
/Zs $(CXXFLAGS_G)
$(CXXFLAGS_R) /FRobj\PROJECTN7.sbr ..\LIB\PROJECTN7.CPP
<<
!ENDIF

obj\PIC_M7.obj : ..\LIB\PIC_M7.CPP C:\C700\INCLUDE\stdlib.h\
C:\C700\INCLUDE\stdio.h C:\C700\INCLUDE\graph.h
C:\C700\INCLUDE\math.h\
C:\C700\INCLUDE\io.h C:\C700\INCLUDE\fcntl.h
C:\C700\INCLUDE\string.h\
C:\C700\INCLUDE\float.h C:\C700\INCLUDE\malloc.h ..\LIB\phdr.h\
c:\ilya\lib\vicalloc.h ..\LIB\pic_mch7.h C:\C700\INCLUDE\vmemory.h\
..\LIB\projctn7.h
!IF $(DEBUG)
    @$(CXX) @<<obj\$(PROJ).rsp
/c $(CXXFLAGS_G)
$(CXXFLAGS_D) /Foobj\PIC_M7.obj ..\LIB\PIC_M7.CPP
<<
!ELSE
    @$(CXX) @<<obj\$(PROJ).rsp
/c $(CXXFLAGS_G)
$(CXXFLAGS_R) /Foobj\PIC_M7.obj ..\LIB\PIC_M7.CPP
<<
!ENDIF

obj\PIC_M7.sbr : ..\LIB\PIC_M7.CPP C:\C700\INCLUDE\stdlib.h\
C:\C700\INCLUDE\stdio.h C:\C700\INCLUDE\graph.h
C:\C700\INCLUDE\math.h\
C:\C700\INCLUDE\io.h C:\C700\INCLUDE\fcntl.h
C:\C700\INCLUDE\string.h\
C:\C700\INCLUDE\float.h C:\C700\INCLUDE\malloc.h ..\LIB\phdr.h\
c:\ilya\lib\vicalloc.h ..\LIB\pic_mch7.h C:\C700\INCLUDE\vmemory.h\
..\LIB\projctn7.h
!IF $(DEBUG)
    @$(CXX) @<<obj\$(PROJ).rsp

```

```

/Zs $(CXXFLAGS_G)
$(CXXFLAGS_D) /FRobj\PIC_M7.sbr ..\LIB\PIC_M7.CPP
<<
!ELSE
    @$(CXX) @<<obj\$(PROJ).rsp
/Zs $(CXXFLAGS_G)
$(CXXFLAGS_R) /FRobj\PIC_M7.sbr ..\LIB\PIC_M7.CPP
<<
!ENDIF

obj\RES_MCH7.obj : ..\LIB\RES_MCH7.CPP C:\C700\INCLUDE\stdlib.h\
    C:\C700\INCLUDE\vmemory.h ..\LIB\pic_mch7.h ..\LIB\res_mch7.h\
    C:\C700\INCLUDE\graph.h ..\LIB\projctn7.h C:\C700\INCLUDE\math.h
!IF $(DEBUG)
    @$(CXX) @<<obj\$(PROJ).rsp
/c $(CXXFLAGS_G)
$(CXXFLAGS_D) /Foobj\RES_MCH7.obj ..\LIB\RES_MCH7.CPP
<<
!ELSE
    @$(CXX) @<<obj\$(PROJ).rsp
/c $(CXXFLAGS_G)
$(CXXFLAGS_R) /Foobj\RES_MCH7.obj ..\LIB\RES_MCH7.CPP
<<
!ENDIF

obj\RES_MCH7.sbr : ..\LIB\RES_MCH7.CPP C:\C700\INCLUDE\stdlib.h\
    C:\C700\INCLUDE\vmemory.h ..\LIB\pic_mch7.h ..\LIB\res_mch7.h\
    C:\C700\INCLUDE\graph.h ..\LIB\projctn7.h C:\C700\INCLUDE\math.h
!IF $(DEBUG)
    @$(CXX) @<<obj\$(PROJ).rsp
/Zs $(CXXFLAGS_G)
$(CXXFLAGS_D) /FRobj\RES_MCH7.sbr ..\LIB\RES_MCH7.CPP
<<
!ELSE
    @$(CXX) @<<obj\$(PROJ).rsp
/Zs $(CXXFLAGS_G)
$(CXXFLAGS_R) /FRobj\RES_MCH7.sbr ..\LIB\RES_MCH7.CPP
<<
!ENDIF

obj\COR_FNC.obj : COR_FNC.CPP C:\C700\INCLUDE\stdio.h
    C:\C700\INCLUDE\stdlib.h\
    C:\C700\INCLUDE\conio.h C:\C700\INCLUDE\float.H\
    C:\C700\INCLUDE\graph.h cor_fnc.h ..\LIB\pic_mch7.h
    ..\LIB\res_mch7.h\
    C:\C700\INCLUDE\vmemory.h ..\LIB\projctn7.h
    C:\C700\INCLUDE\math.h
!IF $(DEBUG)
    @$(CXX) @<<obj\$(PROJ).rsp

```

```
/c $(CXXFLAGS_G)
$(CXXFLAGS_D) /Foobj\COR_FNC.obj COR_FNC.CPP
<<
```

```
!ELSE
```

```
    @$(CXX) @<<obj\$(PROJ).rsp
/c $(CXXFLAGS_G)
$(CXXFLAGS_R) /Foobj\COR_FNC.obj COR_FNC.CPP
<<
```

```
!ENDIF
```

```
obj\COR_FNC.sbr : COR_FNC.CPP C:\C700\INCLUDE\stdio.h
C:\C700\INCLUDE\stdlib.h\
```

```
    C:\C700\INCLUDE\conio.h C:\C700\INCLUDE\float.H\
```

```
    C:\C700\INCLUDE\graph.h cor_fnc.h ..\LIB\pic_mch7.h
```

```
..\LIB\res_mch7.h\
```

```
    C:\C700\INCLUDE\vmemory.h ..\LIB\projctn7.h
```

```
C:\C700\INCLUDE\math.h
```

```
!IF $(DEBUG)
```

```
    @$(CXX) @<<obj\$(PROJ).rsp
/Zs $(CXXFLAGS_G)
$(CXXFLAGS_D) /FRobj\COR_FNC.sbr COR_FNC.CPP
<<
```

```
!ELSE
```

```
    @$(CXX) @<<obj\$(PROJ).rsp
/Zs $(CXXFLAGS_G)
$(CXXFLAGS_R) /FRobj\COR_FNC.sbr COR_FNC.CPP
<<
```

```
!ENDIF
```

```
obj\COR_WIN.obj : COR_WIN.CPP C:\C700\INCLUDE\vmemory.h\
C:\C700\INCLUDE\stdio.h C:\C700\INCLUDE\stdlib.h\
C:\C700\INCLUDE\conio.h C:\C700\INCLUDE\malloc.h\
C:\C700\INCLUDE\float.H C:\C700\INCLUDE\graph.h\
C:\C700\INCLUDE\string.h LNS_CORR.h cor_fnc.h ..\LIB\projctn7.h\
..\LIB\pic_mch7.h ..\LIB\res_mch7.h c:\ilya\lib\lin_tm7.h\
C:\C700\INCLUDE\math.h
```

```
!IF $(DEBUG)
```

```
    @$(CXX) @<<obj\$(PROJ).rsp
/c $(CXXFLAGS_G)
$(CXXFLAGS_D) /Foobj\COR_WIN.obj COR_WIN.CPP
<<
```

```
!ELSE
```

```
    @$(CXX) @<<obj\$(PROJ).rsp
/c $(CXXFLAGS_G)
$(CXXFLAGS_R) /Foobj\COR_WIN.obj COR_WIN.CPP
<<
```

```
!ENDIF
```

```
obj\COR_WIN.sbr : COR_WIN.CPP C:\C700\INCLUDE\vmemory.h\
```

```

C:\C700\INCLUDE\stdio.h C:\C700\INCLUDE\stdlib.h\
C:\C700\INCLUDE\conio.h C:\C700\INCLUDE\malloc.h\
C:\C700\INCLUDE\float.H C:\C700\INCLUDE\graph.h\
C:\C700\INCLUDE\string.h LNS_CORR.h cor_fnc.h ..\LIB\projctn7.h\
..\LIB\pic_mch7.h ..\LIB\res_mch7.h c:\ilya\lib\lin_tm7.h\
C:\C700\INCLUDE\math.h
!IF $(DEBUG)
    @$(CXX) @<<obj\$(PROJ).rsp
/Zs $(CXXFLAGS_G)
$(CXXFLAGS_D) /Frobj\COR_WIN.sbr COR_WIN.CPP
<<
!ELSE
    @$(CXX) @<<obj\$(PROJ).rsp
/Zs $(CXXFLAGS_G)
$(CXXFLAGS_R) /Frobj\COR_WIN.sbr COR_WIN.CPP
<<
!ENDIF

obj\$(PROJ).bsc : $(SBRs)
    $(BSCMAKE) @<<
$(BRFLAGS) $(SBRs)
<<

obj\$(PROJ).exe : $(OBJS)
    -$(NMAKEBSC1) MAKEFLAGS=
    -$(NMAKEBSC2) $(NMFLAGS) -f $(PROJFILE) obj\$(PROJ).bsc
!IF $(DEBUG)
    $(LRF) @<<obj\$(PROJ).lrf
$(RT_OBJS: = +^
) $(OBJS: = +^
)
$@
$(MAPFILE_D)
$(LIBS: = +^
) +
$(LLIBS_G: = +^
) +
$(LLIBS_D: = +^
)
$(DEF_FILE) $(LFLAGS_G) $(LFLAGS_D);
<<
!ELSE
    $(LRF) @<<obj\$(PROJ).lrf
$(RT_OBJS: = +^
) $(OBJS: = +^
)
$@
$(MAPFILE_R)

```

```
$(LIBS: = +^
) +
$(LLIBS_G: = +^
) +
$(LLIBS_R: = +^
)
$(DEF_FILE) $(LFLAGS_G) $(LFLAGS_R);
<<
!ENDIF
    $(LINKER) @obj\$(PROJ).lrf
```

```
.cpp.obj :
!IF $(DEBUG)
    @$(CXX) @<<obj\$(PROJ).rsp
/c $(CXXFLAGS_G)
$(CXXFLAGS_D) /Fo$@ $<
<<
!ELSE
    @$(CXX) @<<obj\$(PROJ).rsp
/c $(CXXFLAGS_G)
$(CXXFLAGS_R) /Fo$@ $<
<<
!ENDIF
```

```
.cpp.sbr :
!IF $(DEBUG)
    @$(CXX) @<<obj\$(PROJ).rsp
/Zs $(CXXFLAGS_G)
$(CXXFLAGS_D) /FR$@ $<
<<
!ELSE
    @$(CXX) @<<obj\$(PROJ).rsp
/Zs $(CXXFLAGS_G)
$(CXXFLAGS_R) /FR$@ $<
<<
!ENDIF
```

```
run: obj\$(PROJ).exe
    obj\$(PROJ).exe $(RUNFLAGS)
```

```
debug: obj\$(PROJ).exe
    CV $(CVFLAGS) obj\$(PROJ).exe $(RUNFLAGS)
```

```

//      Module calculates correlation functions of PROTO_1 and set of
// prototypes. Set of prototypes' names is defined by a MASK correspondes
// to names generated by MAKEPRB and has next structure:
//      [path]&RRW_P.rgb
// Where
//      [path] - optional name of directory;
//      &      - first letter of file name
//      RR      - two digits corresponding to prototype's height
//              (RR= 16| 32 | 48 | 64)
//      W      - number corresponding to window number (see
MAKEPRB
//      description.
//      P      prototype Number
// MASK includes ONLY [path]&RRW_      and programme will
// calculate correlation functions for prototypes with P from 0 to
// first not existing number.

// COMMAND STRING
//
// Ins_corr <PROTO_1_Name> <MASK> [CommandFile]
//
//      <PROTO_1_Name>      File name of PROTOTYPE without
extention
//      <MASK>      Mask for prototypes FileNames without extention
and
//      Prototype's number.
//      [CommandFile]      Optional ASCII file with a run time parameters.
//
// INPUT
//      RGB files of prototypes and corresponding .SGN files created by
// module MAKEPRB.
// RUN TIME parameters:
//
//      0 0 0 0      -shift for all cases have to be 0
//      <CalorSpace>
//      We have used 1 - as worked only with a luminance
//      <Window width>
//      We have used 8
//SEE ALSO FILE "LNS_CORR.INI"
// OUTPUT
//      Correlation functions in PROTO_1.DBC file.

#include <stdlib.h>
#include <conio.h>
#include <stdio.h>
#include <string.h>
#include <graph.h>
#include <float.h>

```







```

#include <fstream.h>
int ReadStrInf(char *name, short *StD)
{ifstream InpF;
 char a[80];
 strcat(strcpy(a,name),".str");
 short i;
 InpF.open(a,ios::in|ios::nocreate);
 if(InpF.fail())
 {InpF.clear(0);
  return 1;
 }
 InpF.getline(a,80,'\n');
 InpF.getline(a,80,'\n');
 InpF.getline(a,80,'\n');
 InpF.getline(a,80,'\n');
 InpF.getline(a,80,'\n');
 InpF.getline(a,80,'\n');
 i=0;
 do
 {
  InpF>>StD[i++];
  if (InpF.eof()|| i>17)
  { StD[-i]=-1;
   break;
  }
  InpF>>StD[i++];
 }
 while(1);
 InpF.close();
 return 0;
}

```

```

//=====
=====
SCR_PNT winsize;
//=====
=====

```

```

int main(int argc, char* argv[])
{int FLG_F=0, FLG_WRIGHT=0;
 FILE *datainf;
 short winstep, map_std;
 short n=0;
 SCR_PNT t_pos;
 if((argc != 3) && (argc !=4))
 {
  printf(" target-file proto_file_mask \n");
  FLG_F=0;
  return(1);
 }
}

```

```

else
    if(argc ==4)
        {FLG_F=1;
        if(!(datainf=fopen(argv[3],"r")) return 0;
        }
    if(FLG_F)
        {get_shift_f(datainf,sh); // GET SCALE
        get_number_3_f(datainf); // GET SCALE
        }
    else
        {get_shift(sh); // GET SCALE 0
        get_number_3();
        }

strcpy(ARGV_1,argv[1]);
PROTOCOL_NAME=argv[1];
PROTOCOL_NAME+=".dbc";
init_protocol(argv[1]);

// ===== GRAPHICS START
    if(GRAPHICS_START(&vc,GRAPHMODE)) exit(-1);
//===== TARGET PICTURE name and vertexes
SCR_PNT target_pos; // CONSTRUCTOR default 0,0
short StrDescr[17];
    _splitpath( argv[1], drive,dir,TAG_name,ext );
    pict_target=sign_storage_rgb(argv[1],vc );
    if(ReadStrInf(argv[1],StrDescr))
        {printf("STR PROTO not exist"); GRAPH_OUT(-1);return -1;
        };

    winsize.r=pict_target.s_rows;
    winstep=winsize.c;
//"PROTO_File\FRAME_Name\STRING_name\SIGN_name\SLength\WinLengt
h\SPACE\n";
fprintf(PROTOCOL,"%s %8s\t%6s\t%12s\t%4d\t%4d\t%12s\n%s",
        (const char
*)PROTOCOL_START,FRAME_Name,STRING_name,SIGN_name,
        pict_target.s_cols,winsize.c,SP[MAP],
        (const char *) PROTO_TAG_HEADER);

//===== PROTOTYPE LOOP OVER names
char proto_name[NAME_LENGTH],buff[4];
SCR_PNT proto_pos,z;
//loop over masks
//return file name without extention in "name" and TRUE 1 if file exist;
short proto_number=0; // # 0;
while( next_pict( proto_name,argv[2],".rgb", proto_number))
    { proto_number++; //next;
    SCR_PNT pr_v[4];

```

```

// PROTO INFORMATION IN PROTOCOL
    pict_proto=sign_storage_rgb(proto_name,vc );
    picture_inf_num_2(proto_name,pr_v,0,".str"); //only for SIGN_name
// "TAG_File\tFRAME_Name\tSTRING_name\tS_Name\tLegnth\n"
    fprintf(PROTOCOL,"%12s\t %8s\t %6s\t %12s\t%4d\n",

proto_name,FRAME_Name,STRING_name,SIGN_name,pict_proto.s_cols);

    TAG_hight=pict_proto.s_rows;
// TARGET PRESENTATION
    _clearscreen(_GCLEARSCREEN);
    proto_pos.c=target_pos.c=10;
    proto_pos.r=(target_pos.r=10)+pict_target.s_rows+5;
    sign_present_RGB( pict_target,target_pos);
    sign_present_RGB(pict_proto,proto_pos);
//=====
    corr_win_proto(pict_target, pict_proto,
                    winsize, winstep,CORR_THRESH,StrDescr);

    pict_proto.free_PCT();
}
_displaycursor( _GCURSORON );
_setvideomode( _DEFAULTMODE );
pict_target.free_PCT();
fclose(PROTOCOL);
return(0);
}

//
=====
=====
void get_shift_f(FILE *f,double * sh) // INITIALISATION GRAPHICMODE,
GET SCALE
{int i;
    for(i=0;i<4; i++)
    {
        fscanf(f,"%lf %lf\n",sh+i++,sh+i);
    }
}

//
=====
=====
void get_shift(double * sh) // INITIALISATION GRAPHICMODE, GET SCALE
{int i;
    cout<< "vertexes shift over rows ( top_right, bottom_right, bottom_left,
top_left \n";
    for (i=0;i<4; i++)
        cin>>sh[i];
}

```

190

```

        for(i=0;i<4;i++)
            if(fscanf(datfp,"%d %d\n",&(vertexes[i].c),&(vertexes[i].r))==EOF)
                {fclose (datfp); return 0;}
        fclose(datfp);
    return 1;
}
//=====
/*void write_sign_inf(char *pr,PCT pict_now)
{ char fl_fp[50],f_name[9];

    int FLG;
    FILE *dathere,*database;
    _splitpath( pr, drive,dir,f_name,ext );
    strcat(strcpy(fl_fp,pr),".sgn");
    dathere=fopen(fl_fp,"w");
    FLG=_access("PROTODB.1", 0 );// -1 if not exist

    if(!(database=fopen("PROTODB.1","a")))
        {strcpy(fl_fp,"CAN NOT CREATE D_BASE FILE");
         exit(-1);}
    fprintf(dathere, " WIN_name FILE_name FRAME_Name STRING_name
SIGN_name\n ");
    fprintf(dathere,"%8s %9s %10s %11s %9s \n",f_name, FILE_name,
FRAME_Name,STRING_name,SIGN_name);
    if(FLG)
        fprintf(database, " WIN_name FILE_name FRAME_Name STRING_name
SIGN_name\n ");
        fprintf(database,"%8s %9s %10s %11s %9s \n",f_name, FILE_name,
FRAME_Name,STRING_name,SIGN_name);
    fprintf(dathere,"%d 0\n",pict_now.s_cols-1);
    fprintf(dathere,"%d %d \n",pict_now.s_cols-1,pict_now.s_rows-1);
    fprintf(dathere," 0 %d\n",pict_now.s_rows-1);
    fprintf(dathere," 0 0\n");

    fclose(dathere);
    fclose(database);
} */
//=====
//=====
int picture_inf_num(char *name,SCR_PNT *vertexes,short n)
{int i,j;
 char new_name[25];
 FILE *datfp;
 strcat(strcpy(new_name,name),".sgn");
 if(!(datfp=fopen(new_name,"r")) return 0;
 fscanf(datfp,"%s\n",new_name);

```

```

        for(j=0;j<n+1;j++)
            for(i=0;i<4;i++)
                if(fscanf(datfp,"%d %d\n",&(vertexes[i].c),&(vertexes[i].r))==EOF)
                    {fclose (datfp); return 0;}
        fclose(datfp);
    return 1;
}
//=====
int picture_inf_num_new(char *name,SCR_PNT *vertexes,short n)
{
    int i,j;
    char new_str[80];
    FILE *datfp;
    int r,FLG=0;
    strcat(strcpy(new_str,name),".sgn");
    if(!(datfp=fopen(new_str,"r"))) return 0;
    r=fscanf(datfp,"%[^\n]s ",new_str);
    r=fscanf(datfp," %[^\n]s ",new_str);
    if(__iscsymf( (int)new_str[0]))//FILE INFORMATION
        //(letter or underscore)
        {sscanf(new_str," %s %s %s %s",&FILE_name, &FRAME_Name,
            &STRING_name, &SIGN_name);
        r=fscanf(datfp," %[^\n]s ",new_str);
        }
    for(j=0;j<n+1;j++)
        for(i=0;i<4;i++)
            {if(FLG)
                if(fscanf(datfp," %[^\n]s",new_str)==EOF)
                    {fclose (datfp); return 0;}
                FLG=1;
                sscanf(new_str," %d %d",&(vertexes[i].c),&(vertexes[i].r));
            }
        fclose(datfp);
    return 1;
}
//=====

```

```

#include <vmemory.h>
#include <stdio.h>
#include <stdlib.h>
#include <conio.h>
#include <malloc.h>
#include <float.h>
#include <graph.h>
#include <string.h>

#include "Ins_corr.h"
//include "tem_plt7.h"
#include "cor_fnc.h"
//include "deb_out.h"

#define MAX_LINE 1024

extern FILE *PROTOCOL;

extern double GAMMA,Thresh_mdl;
extern short MAP;
extern short VOITING;
extern struct _videoconfig vc;
extern char f_name[40],FILE_name[40], FRAME_Name[40],
          STRING_name[40], SIGN_name[40];
//=====
===
void draw_int(short st,short w,COLOR_VEC intr);
//=====
===
void draw_color_corr_1(COLOR_VEC corr,short F,
short CH_HIGHT,
short CH_BASE,double THRESH,
short pos_now,short size);
//=====
void draw_chart(double *dist_line,short n,double max_value=0,
short CH_HIGHT= 100,
short CH_BASE= 480,double THRESH=0,
short t_pos=60);
//=====
=====
#define HOR_HARM 2
#define VERT_HARM 4
//_____
inline COLOR_VEC sign_for_col(short d, COLOR_VEC col)
{ COLOR_VEC out;
  int i;
  for (i=0;i<3;i++)
    out.c[i]= d?col.c[i]:-col.c[i];
  return out;
}

```

```

    }
//-----
/* COLOR_VEC int_value_1(PCT w,double Thr,
   COLOR_VEC (*p_funct)(COLOR_RGB p1,double
   Thresh_md1),AVERAGE_VEC w_av)
{COLOR_VEC col,sum[9][9],out,c1;
  const COLOR_VEC z={0,0,0};

  short h_t,v_t,i,x,y,h,v,
  half_x=w.s_cols>>1,half_y=w.s_rows>>1,
  quot_x=w.s_cols>>2,quot_y=w.s_rows>>2;
  long n;

  for(h=0;h<HOR_HARM+1;h++)
    for(v=0;v<VERT_HARM+1;v++)
      sum[v][h].c[0]=sum[v][h].c[1]=sum[v][h].c[2]=0.0;
  n=w.s_cols*w.s_rows;
  n*=n;
  for(y=0;y<w.s_rows;y++)
    for(v=0;v<VERT_HARM+1;v++)
      {
        v_t=y*((v+1)>>1);
        v_t=(v_t+(v & 0x0001 ? quot_y:0))/half_y;
        v_t &= 0x0001;
        for(x=0;x<w.s_cols;x++)
          {col=p_funct(w.get_pixel(y,x),Thr);
            c1=sign_for_col(v_t,col);
            for(h=0;h<HOR_HARM+1;h++)
              {
                h_t=x*((h+1)>>1);
                h_t=(h_t+(h & 0x0001 ? quot_x:0))/half_x;
                h_t &= 0x0001;
                c1=sign_for_col(h_t,c1);
                for(i=0;i<3;i++)
                  sum[v][h].c[i]+=c1.c[i];
              }
            }
      }
  }
  double s0,dd,max_v=0,th;
  for(dd=i=0;i<3;i++)
    {for(s0=h=0;h<HOR_HARM+1;h++)
      for(v=0;v<VERT_HARM+1;v++)
        if(h||v)
          s0+=sum[v][h].c[i]*sum[v][h].c[i];
      s0/=n;
      dd=w_av.var[i]+w_av.md[i]*w_av.md[i];
      out.c[i]=(dd?s0/dd:1);
      max_v=(max_v<out.c[i])?out.c[i]:max_v;
    }
}

```



```

    for(i=0;i<3;i++)
        {th=out.c[i]/max_v;
        //          THRESHOLDING
        if(th<0.2)
            out.c[i]=0;
        }
    return out;
} */
//=====
=====
COLOR_VEC (*PointColFunct())(COLOR_RGB p1,double Thresh_mdl)

{ switch ( MAP)
    {case NTSC:return(color_space_NTSC);
    case New_plan:return(color_space_NEW);
    case HSI:return(color_space_RGB);
    case RGB:return(color_space_RGB_simple);
    case LUMIN_THR:return(color_space_LUMIN_THR);
    case IHS:return(color_space_IHS);
    };
    return NULL;
}
//=====
=====
const short CH_HIGHT_D=100, CH_BASE_D=470,
            CH_HIGHT=100, CH_BASE=450, t_pos=40;
//=====
double scale_fact=1;
//=====
void corr_win_proto(PCT win_source,PCT Proto, SCR_PNT win_size
                    ,short win_step,double CORR_THRESH,short *StripEnds)
{
    short i;
    char mess[40];
    short F=0;
    COLOR_VEC (*p_funct)(COLOR_RGB p1,double Thresh_mdl);
    p_funct=PointColFunct();

    PCT win(win_size.c,win_size.r);
    PCT tag(win_size.c,win_size.r);
    SCR_PNT st_t,st_win;
    AVERAGE_VEC middle_win[64],middle_tag;
    const AVERAGE_VEC z={{0,0,0},{0,0,0}};

    COLOR_VEC *corr_now,cr;
    const COLOR_VEC z_col={0.0,0.0,0.0};
    int line_size=win_source.s_cols+Proto.s_cols;
    //memory allocation

```

```

    if((corr_now= (COLOR_VEC*) malloc(
sizeof(COLOR_VEC)*(size_t)line_size*3))==NULL)
        {printf("WIN NOT MEMORY"); return;};

    st_t.r=0;
double dd;
    st_win.r=0;
    short k,FLG_COL=1;
    short StripStart,StripStop;
    short PartNum;
    k=PartNum=0;
    while(StripEnds[PartNum]>=0)
        {StripStart=StripEnds[PartNum++];
        StripStop=StripEnds[PartNum++];
        for (st_win.c=StripStart;
st_win.c+win_size.c<=StripStop;st_win.c+=win_step,k++)
            {
                FLG_COL=1;
                for(i=0;i<line_size;corr_now[i++]=z_col);

                win.load_template(win_source,st_win);
                middle_win[k]=average(win,Thresh_mdl,p_funct);

#ifdef MICI
#endif

const COLOR_VEC z_UNIT={1.0,1.0,1.0};

        for (st_t.c=0;st_t.c<=Proto.s_cols-win_size.c;st_t.c++)
            {
//=====
                tag.load_template(Proto,st_t);
                middle_tag=average(tag,Thresh_mdl,p_funct);
// DIFF ABS VALUES
#ifdef SINGL_VAL
                cr=template_conv_1( tag,win,Thresh_mdl,z_UNIT,p_funct);
                strcpy(mess," VECTOR Approach to CORRELATION ");
                corr_now[st_t.c]=Correlation_single_1(cr,middle_tag,middle_win[k],z_
UNIT);
#ifdef ABS_VALUE
                strcpy(mess," DIFF ABS VALUES/ max ABS VALUES");
                cr=
                    template_abs_diff_1 (tag,win,Thresh_mdl,z_UNIT,p_funct,
                    middle_tag,middle_win[k]);
#else
                cr=template_conv_1( tag,win,Thresh_mdl,z_UNIT,p_funct);
                strcpy(mess," PEARSON CORR. ");
                corr_now[st_t.c]=Correlation(cr,middle_tag,middle_win[k],z_UNIT);
#endif
            }
#endif

```

```

#endif

// ONLY LUMINANCE
//      strcat(mess," ALL 3 COMP");
//      strcat(mess," Only 0 COMP");
//      corr_now[st_t.c].c[1]=corr_now[st_t.c].c[2]=
//          corr_now[st_t.c].c[0];

#ifdef MICI
draw_color_corr_1(
corr_now[st_t.c],FLG_COL,CH_HIGHT_D,CH_BASE_D,CORR_THRESH,
st_t.c,Proto.s_cols);

FLG_COL=0;
#endif
    }

//=====FILL PROTOCOL
//$ WILL BE USED AS SEPARATOR FOR READING
fprintf(PROTOCOL,"%t%st%4d\t $n",mess, st_win.c);
for(i=0;i<Proto.s_cols;i++) //ONLY 0 COMP
    fprintf(PROTOCOL,"%6g\t",corr_now[i].c[0]);
    fprintf(PROTOCOL," \n");
}
}

win.free_PCT();
tag.free_PCT();
free((void *)corr_now);
return ;
}

//=====
=====

//=====
=====

void draw_chart(double *dist_line,short n,double max_value,
short CH_HIGHT,
short CH_BASE,double THRESH,
short t_pos)

{short i,j;
double p,
crit=max_value;
if(!max_value)
    for (i=0;i<n;i++)
        crit=(dist_line[i]>crit)? dist_line[i]:crit;
else crit=max_value;
if(!crit)
    crit=1;
p= CH_HIGHT*(1-THRESH/crit);
_moveto( 0,CH_BASE-CH_HIGHT    );

```

```

    _lineto(n,CH_BASE-CH_HEIGHT);
    _moveto(0,CH_BASE-(short)p);
    _lineto(n,CH_BASE-(short)p);
    _moveto( (short) 0,(short) CH_BASE );
    for (i=0;i<__min(n,vc.numxpixels);i++)
        {j=CH_BASE-CH_HEIGHT+(short)(dist_line[i]*CH_HEIGHT/crit);
        if(j<0)
        //    getch()
            ;
        else
            if(!_lineto( i,j))
        //            getch()
            ;
        }
    if(t_pos)
        _settextposition( t_pos,30);
    char buffer[30];
    sprintf(buffer, "MAX = %f10 ", crit );
    _outtext( buffer );

}
//=====
=
void draw_int(short st,short w,COLOR_VEC intr)
{short CH_HEIGHT=100,
  CH_BASE=200;
double p;
    _setcolor( color_num(240,240,240));
    _setcolor( color_num(240,0,0));
    p=CH_BASE-CH_HEIGHT*intr.c[0];
    _moveto( st,(short)p);
    _lineto(st+w,(short)p);
    _setcolor( color_num(0,240,0));
    p=CH_BASE-CH_HEIGHT*intr.c[1];
    _moveto( st,(short)p);
    _lineto(st+w,(short)p);
    _setcolor( color_num(0,0,240));
    p=CH_BASE-CH_HEIGHT*intr.c[2];
    _moveto( st,(short)p);
    _lineto(st+w,(short)p);
}
//=====
=====
void draw_color_corr(COLOR_VEC corr,COLOR_RGB *corr_old,short F,
short CH_HEIGHT,
short CH_BASE,double THRESH,
short pos_now)
{double p;
short j;

```

```

if(F)
{
    _setcolor( color_num(240,240,240));
    p= CH_HEIGHT*(1-THRESH);
    _moveto( 0,CH_BASE-CH_HEIGHT );
    _lineto(512,CH_BASE-CH_HEIGHT);
    _moveto(0,CH_BASE-(short)p);
    _lineto(512,CH_BASE-(short)p);
}
_setcolor( color_num(240,240,240));
_moveto( pos_now,corr_old->r);
j=CH_BASE-CH_HEIGHT+(short)(corr.c[0]*CH_HEIGHT);
_lineto(pos_now+1 ,j);
corr_old->r=j;
_moveto( pos_now,corr_old->g);
_setcolor( color_num(240,0,0));
j=CH_BASE-CH_HEIGHT+(short)(corr.c[1]*CH_HEIGHT);
_lineto(pos_now+1 ,j);
corr_old->g=j;
_moveto( pos_now,corr_old->b);
_setcolor( color_num(0,240,0));
j=CH_BASE-CH_HEIGHT+(short)(corr.c[2]*CH_HEIGHT);
_lineto(pos_now+1 ,j);
corr_old->b=j;
}
//=====
=====
void draw_color_corr_1(COLOR_VEC corr,short F,
short CH_HEIGHT,
short CH_BASE,double THRESH,
short pos_now,short size)
{
    short j,k,l,i,st;
    static short real_size,pos_old;
    short POS;
    static COLOR_RGB corr_old;
    real_size=size*scale_fact;
    POS=10+pos_now*scale_fact;
    _setcolor( color_num(240,240,240));
    if(F)
    {
        _setcolor( color_num(0,0,0));
        _rectangle( _GFILLINTERIOR,0,CH_BASE-3*CH_HEIGHT-
(CH_HEIGHT_D>>1),
                    real_size ,CH_BASE);
        _setcolor( color_num(240,240,240));
        corr_old.r=k=CH_BASE-2*CH_HEIGHT-40;
        st=CH_HEIGHT/10;
        for(i=0;i<3;i++)

```

```

{
    _moveto( 10,k-CH_HEIGHT);
    _lineto(10,k);
    _lineto(10+real_size,k);
    _moveto(10,k-CH_HEIGHT*THRESH);
    _lineto(10+real_size,k-CH_HEIGHT*THRESH);
    for(l=0,j=1;j<11;j++)
    {l+=st;
      _moveto(
        (j==5)?5:((j==10)?0:7)
        ,k-l);
      _lineto(10,k-l);
    }
    k+=(CH_HEIGHT+20);
}
corr_old.g=corr_old.r+CH_HEIGHT+20;
corr_old.b=corr_old.g+CH_HEIGHT+20;
pos_old=10;
}
_setcolor( color_num(240,240,240));
k=CH_BASE;
_moveto( pos_old,corr_old.b);
j=k-(short)(corr.c[2]*CH_HEIGHT);
_lineto((short)(POS),j);
corr_old.b=j;

k=(CH_HEIGHT+20);
_moveto( pos_old,corr_old.g);
j=k-(short)(corr.c[1]*CH_HEIGHT);
_lineto((short)(POS),j);
corr_old.g=j;

k=(CH_HEIGHT+20);
_moveto( pos_old,corr_old.r);
j=k-(short)(corr.c[0]*CH_HEIGHT);
_lineto((short)(POS),j);
corr_old.r=j;
pos_old=POS;
}

```

```
void corr_win_proto(PCT win_source, PCT Proto, SCR_PNT win_size  
    , short win_step, double CORR_THRESH,  
    short *StripEnds) ;
```

```

ORIGIN = PWB
ORIGIN_VER = 2.0
PROJ = PRT_ANL2
PROJFILE = PRT_ANL2.MAK
BUILDDIR = obj
DEBUG = 1

```

```

BRFLAGS = /o obj\$(PROJ).bsc
BSCMAKE = bscmake
SBRPACK = sbrpack
NMAKEBSC1 = set
NMAKEBSC2 = nmake
BROWSE = 1
CC = cl
CFLAGS_G = /W2 /BATCH /FR$*.sbr
CFLAGS_D = /f /Zi /Od
CFLAGS_R = /f- /Ot /Oi /Ol /Oe /Og /Gs
CXX = cl
CXXFLAGS_G = /AL /W4 /G2 /D_DOS /BATCH /FR$*.sbr
CXXFLAGS_D = /f- /Od /FPi87 /Zi /DMICI /DSINGLE_WIN
CXXFLAGS_R = /f- /Ot /Ol /Og /Oe /Oi /FPi87 /Gs /DMICI /DSINGLE_WIN
MAPFILE_D = NUL
MAPFILE_R = NUL
LFLAGS_G = /NOI /STACK:32000 /BATCH /ONERROR:NOEXE
LFLAGS_D = /CO /FAR /PACKC
LFLAGS_R = /EXE /FAR /PACKC
LINKER = link
ILINK = ilink
LRF = echo > NUL
ILFLAGS = /a /e
LLIBS_G = graphics lafxc
CVFLAGS = /25 /S
RUNFLAGS = ..\win4\s160_0 corrrhr.06
PACK_SBRS = 1

```

```

FILES = PRT_ANL2.CPP ..\LIB\VICALLOC.CPP ..\LIB\PROJECTN7.CPP\
        ..\LIB\PIC_M7.CPP ..\LIB\RES_MCH7.CPP PRT_2MD.CPP
OBJS = obj\PRT_ANL2.obj obj\VICALLOC.obj obj\PROJECTN7.obj
        obj\PIC_M7.obj
        obj\RES_MCH7.obj obj\PRT_2MD.obj
SBRS = obj\PRT_ANL2.sbr obj\VICALLOC.sbr obj\PROJECTN7.sbr
        obj\PIC_M7.sbr
        obj\RES_MCH7.sbr obj\PRT_2MD.sbr

```

```
all: obj\$(PROJ).exe
```

```

.SUFFIXES:
.SUFFIXES:
.SUFFIXES: .obj .sbr .cpp

```



```

obj\PRT_ANL2.obj : PRT_ANL2.CPP C:\C700\INCLUDE\stdlib.h\
C:\C700\INCLUDE\conio.h C:\C700\INCLUDE\stdio.h\
C:\C700\INCLUDE\string.h C:\C700\INCLUDE\graph.h\
C:\C700\INCLUDE\float.H C:\C700\INCLUDE\time.h
C:\C700\INCLUDE\ctype.h\
C:\C700\INCLUDE\fstream.h C:\C700\MFC\INCLUDE\afx.h prt_anls.h\
PIC_PRO.h C:\C700\INCLUDE\iostream.h
C:\C700\MFC\INCLUDE\afx.inl\
..LIB\projctn7.h ..LIB\pic_mch7.h ..LIB\res_mch7.h\
c:\ilya\lib\lin_tm7.h C:\C700\INCLUDE\direct.h
C:\C700\INCLUDE\ios.h\
C:\C700\INCLUDE\streamb.h C:\C700\INCLUDE\istream.h\
C:\C700\INCLUDE\ostream.h C:\C700\INCLUDE\math.h\
C:\C700\INCLUDE\vmemory.h
!IF $(DEBUG)
    @$(CXX) @<<obj\$(PROJ).rsp
/c $(CXXFLAGS_G)
$(CXXFLAGS_D) /Foobj\PRT_ANL2.obj PRT_ANL2.CPP
<<
!ELSE
    @$(CXX) @<<obj\$(PROJ).rsp
/c $(CXXFLAGS_G)
$(CXXFLAGS_R) /Foobj\PRT_ANL2.obj PRT_ANL2.CPP
<<
!ENDIF

obj\PRT_ANL2.sbr : PRT_ANL2.CPP C:\C700\INCLUDE\stdlib.h\
C:\C700\INCLUDE\conio.h C:\C700\INCLUDE\stdio.h\
C:\C700\INCLUDE\string.h C:\C700\INCLUDE\graph.h\
C:\C700\INCLUDE\float.H C:\C700\INCLUDE\time.h
C:\C700\INCLUDE\ctype.h\
C:\C700\INCLUDE\fstream.h C:\C700\MFC\INCLUDE\afx.h prt_anls.h\
PIC_PRO.h C:\C700\INCLUDE\iostream.h
C:\C700\MFC\INCLUDE\afx.inl\
..LIB\projctn7.h ..LIB\pic_mch7.h ..LIB\res_mch7.h\
c:\ilya\lib\lin_tm7.h C:\C700\INCLUDE\direct.h
C:\C700\INCLUDE\ios.h\
C:\C700\INCLUDE\streamb.h C:\C700\INCLUDE\istream.h\
C:\C700\INCLUDE\ostream.h C:\C700\INCLUDE\math.h\
C:\C700\INCLUDE\vmemory.h
!IF $(DEBUG)
    @$(CXX) @<<obj\$(PROJ).rsp
/Zs $(CXXFLAGS_G)
$(CXXFLAGS_D) /FRobj\PRT_ANL2.sbr PRT_ANL2.CPP
<<
!ELSE
    @$(CXX) @<<obj\$(PROJ).rsp
/Zs $(CXXFLAGS_G)

```

```

$(CXXFLAGS_R) /FRobj\VRT_ANL2.sbr VRT_ANL2.CPP
<<
!ENDIF

obj\VICALLOC.obj : ..\LIB\VICALLOC.CPP C:\C700\INCLUDE\stdlib.h\
C:\C700\INCLUDE\stdio.h C:\C700\INCLUDE\vmemory.h\
C:\C700\INCLUDE\malloc.h
!IF $(DEBUG)
    @$(CXX) @<<obj\$(PROJ).rsp
/c $(CXXFLAGS_G)
$(CXXFLAGS_D) /Foobj\VICALLOC.obj ..\LIB\VICALLOC.CPP
<<
!ELSE
    @$(CXX) @<<obj\$(PROJ).rsp
/c $(CXXFLAGS_G)
$(CXXFLAGS_R) /Foobj\VICALLOC.obj ..\LIB\VICALLOC.CPP
<<
!ENDIF

obj\VICALLOC.sbr : ..\LIB\VICALLOC.CPP C:\C700\INCLUDE\stdlib.h\
C:\C700\INCLUDE\stdio.h C:\C700\INCLUDE\vmemory.h\
C:\C700\INCLUDE\malloc.h
!IF $(DEBUG)
    @$(CXX) @<<obj\$(PROJ).rsp
/Zs $(CXXFLAGS_G)
$(CXXFLAGS_D) /FRobj\VICALLOC.sbr ..\LIB\VICALLOC.CPP
<<
!ELSE
    @$(CXX) @<<obj\$(PROJ).rsp
/Zs $(CXXFLAGS_G)
$(CXXFLAGS_R) /FRobj\VICALLOC.sbr ..\LIB\VICALLOC.CPP
<<
!ENDIF

obj\PROJECTN7.obj : ..\LIB\PROJECTN7.CPP C:\C700\INCLUDE\graph.h\
C:\C700\INCLUDE\stdlib.h C:\C700\INCLUDE\iostream.h
..\LIB\projctn7.h\
C:\C700\INCLUDE\ios.h C:\C700\INCLUDE\streamb.h\
C:\C700\INCLUDE\istream.h C:\C700\INCLUDE\ostream.h\
C:\C700\INCLUDE\math.h
!IF $(DEBUG)
    @$(CXX) @<<obj\$(PROJ).rsp
/c $(CXXFLAGS_G)
$(CXXFLAGS_D) /Foobj\PROJECTN7.obj ..\LIB\PROJECTN7.CPP
<<
!ELSE
    @$(CXX) @<<obj\$(PROJ).rsp
/c $(CXXFLAGS_G)
$(CXXFLAGS_R) /Foobj\PROJECTN7.obj ..\LIB\PROJECTN7.CPP

```

&lt;&lt;

!ENDIF

```
obj\PROJECTN7.sbr : ..\LIB\PROJECTN7.CPP C:\C700\INCLUDE\graph.h
C:\C700\INCLUDE\stdlib.h C:\C700\INCLUDE\iostream.h
..\LIB\projctn7.h
C:\C700\INCLUDE\ios.h C:\C700\INCLUDE\streamb.h
C:\C700\INCLUDE\istream.h C:\C700\INCLUDE\ostream.h
C:\C700\INCLUDE\math.h
```

!IF \$(DEBUG)

@\$(CXX) @&lt;&lt;obj\\$(PROJ).rsp

/Zs \$(CXXFLAGS\_G)

\$(CXXFLAGS\_D) /Frobj\PROJECTN7.sbr ..\LIB\PROJECTN7.CPP

&lt;&lt;

!ELSE

@\$(CXX) @&lt;&lt;obj\\$(PROJ).rsp

/Zs \$(CXXFLAGS\_G)

\$(CXXFLAGS\_R) /Frobj\PROJECTN7.sbr ..\LIB\PROJECTN7.CPP

&lt;&lt;

!ENDIF

```
obj\PIC_M7.obj : ..\LIB\PIC_M7.CPP C:\C700\INCLUDE\stdlib.h
C:\C700\INCLUDE\stdio.h C:\C700\INCLUDE\graph.h
C:\C700\INCLUDE\math.h
C:\C700\INCLUDE\io.h C:\C700\INCLUDE\fcntl.h
C:\C700\INCLUDE\string.h
C:\C700\INCLUDE\float.h C:\C700\INCLUDE\malloc.h ..\LIB\phdr.h
..\LIB\vicalloc.h ..\LIB\pic_mch7.h C:\C700\INCLUDE\vmemory.h
..\LIB\projctn7.h
```

!IF \$(DEBUG)

@\$(CXX) @&lt;&lt;obj\\$(PROJ).rsp

/c \$(CXXFLAGS\_G)

\$(CXXFLAGS\_D) /Foobj\PIC\_M7.obj ..\LIB\PIC\_M7.CPP

&lt;&lt;

!ELSE

@\$(CXX) @&lt;&lt;obj\\$(PROJ).rsp

/c \$(CXXFLAGS\_G)

\$(CXXFLAGS\_R) /Foobj\PIC\_M7.obj ..\LIB\PIC\_M7.CPP

&lt;&lt;

!ENDIF

```
obj\PIC_M7.sbr : ..\LIB\PIC_M7.CPP C:\C700\INCLUDE\stdlib.h
C:\C700\INCLUDE\stdio.h C:\C700\INCLUDE\graph.h
C:\C700\INCLUDE\math.h
C:\C700\INCLUDE\io.h C:\C700\INCLUDE\fcntl.h
C:\C700\INCLUDE\string.h
C:\C700\INCLUDE\float.h C:\C700\INCLUDE\malloc.h ..\LIB\phdr.h
..\LIB\vicalloc.h ..\LIB\pic_mch7.h C:\C700\INCLUDE\vmemory.h
..\LIB\projctn7.h
```

```

IIF $(DEBUG)
    @$(CXX) @<<obj\$(PROJ).rsp
/Zs $(CXXFLAGS_G)
$(CXXFLAGS_D) /FRobj\PIC_M7.sbr ..\LIB\PIC_M7.CPP
<<
ELSE
    @$(CXX) @<<obj\$(PROJ).rsp
/Zs $(CXXFLAGS_G)
$(CXXFLAGS_R) /FRobj\PIC_M7.sbr ..\LIB\PIC_M7.CPP
<<
ENDIF

obj\RES_MCH7.obj : ..\LIB\RES_MCH7.CPP C:\C700\INCLUDE\stdlib.h\
C:\C700\INCLUDE\vmemory.h ..\LIB\pic_mch7.h ..\LIB\res_mch7.h\
C:\C700\INCLUDE\graph.h ..\LIB\projctn7.h C:\C700\INCLUDE\math.h
IIF $(DEBUG)
    @$(CXX) @<<obj\$(PROJ).rsp
/c $(CXXFLAGS_G)
$(CXXFLAGS_D) /Foobj\RES_MCH7.obj ..\LIB\RES_MCH7.CPP
<<
ELSE
    @$(CXX) @<<obj\$(PROJ).rsp
/c $(CXXFLAGS_G)
$(CXXFLAGS_R) /Foobj\RES_MCH7.obj ..\LIB\RES_MCH7.CPP
<<
ENDIF

obj\RES_MCH7.sbr : ..\LIB\RES_MCH7.CPP C:\C700\INCLUDE\stdlib.h\
C:\C700\INCLUDE\vmemory.h ..\LIB\pic_mch7.h ..\LIB\res_mch7.h\
C:\C700\INCLUDE\graph.h ..\LIB\projctn7.h C:\C700\INCLUDE\math.h
IIF $(DEBUG)
    @$(CXX) @<<obj\$(PROJ).rsp
/Zs $(CXXFLAGS_G)
$(CXXFLAGS_D) /FRobj\RES_MCH7.sbr ..\LIB\RES_MCH7.CPP
<<
ELSE
    @$(CXX) @<<obj\$(PROJ).rsp
/Zs $(CXXFLAGS_G)
$(CXXFLAGS_R) /FRobj\RES_MCH7.sbr ..\LIB\RES_MCH7.CPP
<<
ENDIF

obj\PRT_2MD.obj : PRT_2MD.CPP C:\C700\INCLUDE\io.h
C:\C700\INCLUDE\iostream.h\
C:\C700\INCLUDE\fstream.h prt_anls.h PIC_PRO.h
C:\C700\INCLUDE\ios.h\
C:\C700\INCLUDE\streamb.h C:\C700\INCLUDE\istream.h\
C:\C700\INCLUDE\ostream.h ..\LIB\projctn7.h ..\LIB\pic_mch7.h\
..\LIB\res_mch7.h c:\ilya\lib\lin_tm7.h C:\C700\INCLUDE\stdlib.h\

```

```

C:\C700\INCLUDE\direct.h C:\C700\MFC\INCLUDE\afx.h\
C:\C700\INCLUDE\math.h C:\C700\INCLUDE\vmemory.h\
C:\C700\INCLUDE\graph.h C:\C700\INCLUDE\string.h\
C:\C700\INCLUDE\stdio.h C:\C700\INCLUDE\time.h\
C:\C700\MFC\INCLUDE\afx.inl
!IF $(DEBUG)
    @$(CXX) @<<obj\$(PROJ).rsp
/c $(CXXFLAGS_G)
$(CXXFLAGS_D) /Foobj\PRT_2MD.obj PRT_2MD.CPP
<<
!ELSE
    @$(CXX) @<<obj\$(PROJ).rsp
/c $(CXXFLAGS_G)
$(CXXFLAGS_R) /Foobj\PRT_2MD.obj PRT_2MD.CPP
<<
!ENDIF

obj\PRT_2MD.sbr : PRT_2MD.CPP C:\C700\INCLUDE\io.h
C:\C700\INCLUDE\iostream.h\
C:\C700\INCLUDE\fstream.h prt_anls.h PIC_PRO.h
C:\C700\INCLUDE\ios.h\
C:\C700\INCLUDE\streamb.h C:\C700\INCLUDE\istream.h\
C:\C700\INCLUDE\ostream.h ..\LIB\projctn7.h ..\LIB\pic_mch7.h\
..\LIB\res_mch7.h c:\ilya\lib\lin_tm7.h C:\C700\INCLUDE\stdlib.h\
C:\C700\INCLUDE\direct.h C:\C700\MFC\INCLUDE\afx.h\
C:\C700\INCLUDE\math.h C:\C700\INCLUDE\vmemory.h\
C:\C700\INCLUDE\graph.h C:\C700\INCLUDE\string.h\
C:\C700\INCLUDE\stdio.h C:\C700\INCLUDE\time.h\
C:\C700\MFC\INCLUDE\afx.inl
!IF $(DEBUG)
    @$(CXX) @<<obj\$(PROJ).rsp
/Zs $(CXXFLAGS_G)
$(CXXFLAGS_D) /FRobj\PRT_2MD.sbr PRT_2MD.CPP
<<
!ELSE
    @$(CXX) @<<obj\$(PROJ).rsp
/Zs $(CXXFLAGS_G)
$(CXXFLAGS_R) /FRobj\PRT_2MD.sbr PRT_2MD.CPP
<<
!ENDIF

obj\$(PROJ).bsc : $(SBRs)
    $(BSCMAKE) @<<
$(BRFLAGS) $(SBRs)
<<

obj\$(PROJ).exe : $(OBJS)
    -$(NMAKEBSC1) MAKEFLAGS=

```

```

        -$(NMAKEBSC2) $(NMFLAGS) -f $(PROJFILE) obj\$(PROJ).bsc
!!IF $(DEBUG)
        $(LRF) @<<obj\$(PROJ).lrf
$(RT_OBJS: = +^
) $(OBJS: = +^
)
$@
$(MAPFILE_D)
$(LIBS: = +^
) +
$(LLIBS_G: = +^
) +
$(LLIBS_D: = +^
)
$(DEF_FILE) $(LFLAGS_G) $(LFLAGS_D);
<<
!ELSE
        $(LRF) @<<obj\$(PROJ).lrf
$(RT_OBJS: = +^
) $(OBJS: = +^
)
$@
$(MAPFILE_R)
$(LIBS: = +^
) +
$(LLIBS_G: = +^
) +
$(LLIBS_R: = +^
)
$(DEF_FILE) $(LFLAGS_G) $(LFLAGS_R);
<<
!ENDIF
$(LINKER) @obj\$(PROJ).lrf

```

```

.cpp.obj:
!!IF $(DEBUG)
        @$(CXX) @<<obj\$(PROJ).rsp
/c $(CXXFLAGS_G)
$(CXXFLAGS_D) /Fo$@ $<
<<
!ELSE
        @$(CXX) @<<obj\$(PROJ).rsp
/c $(CXXFLAGS_G)
$(CXXFLAGS_R) /Fo$@ $<
<<
!ENDIF

```

```

.cpp.sbr:

```

```
!IF $(DEBUG)
    @$(CXX) @<<obj\$(PROJ).rsp
/Zs $(CXXFLAGS_G)
$(CXXFLAGS_D) /FR$@ $<
<<
!ELSE
    @$(CXX) @<<obj\$(PROJ).rsp
/Zs $(CXXFLAGS_G)
$(CXXFLAGS_R) /FR$@ $<
<<
!ENDIF

run: obj\$(PROJ).exe
    obj\$(PROJ).exe $(RUNFLAGS)

debug: obj\$(PROJ).exe
    CV $(CVFLAGS) obj\$(PROJ).exe $(RUNFLAGS)
```

```

#ifndef PIC_PRO
#define PIC_PRO
#include <stdlib.h>
#include <direct.h>
#include <afx.h>
#include <pic_mch7.h>
//=====
CString MakeName(char *p );
CString MakeName(CString N);

//=====
const SCR_PNT z_0(0,0);
class PRT:public PCT
{public:
//information
    CString PathName;
    CString FRAME_Number;
    CString STRING_name;
    CString SIGN_name;
    short Pos; // Position in the string
    long NumberOfChk,MaxNum;
    double *Charact;
//models
    PRT::~PRT()
        {this->free_PCT();
        Pos=0;
        if(MaxNum)
            delete Charact;
        Charact=NULL;
        MaxNum=NumberOfChk=0;
        }

//-----
    PRT::PRT()
        {NumberOfChk=MaxNum=s_cols=s_rows=0;
        Charact=NULL;pict=NULL;
        }

//-----
    PRT::PRT (short n_cols, short n_rows)
        {*(PCT *)this=PCT::PCT(n_cols,n_rows);
        NumberOfChk=MaxNum=0;
        Charact=NULL;
        }

//=====
int read_proto_SGN()
{
    CString new_name(' ',80);

    PathName=MakeName(PathName);

```

216



```

    new_name=PathName+".sgn";
    char now[80];
    FILE *datfp;
    if(!(datfp=fopen((const char*)new_name,"r"))) return 1;
        if(fscanf(datfp,"%*[^\\n]s")==EOF)goto ERR;
        if(fscanf(datfp,"%s ",now)==EOF)goto ERR;
        if(fscanf(datfp,"%s ",now)==EOF)goto ERR;
        if(fscanf(datfp,"%s ",now)==EOF)goto
ERR;FRAME_Number=now;
        if(fscanf(datfp,"%s ",now)==EOF)goto
ERR;STRING_name=now;
        if(fscanf(datfp,"%s ",now)==EOF)goto ERR; SIGN_name=now;
        FRAME_Number.MakeUpper();
        STRING_name.MakeUpper();
        SIGN_name.MakeUpper();
        fclose(datfp);
        return 0;
    ERR:fclose (datfp); return 1;
}
//=====
int proto_storage_rgb(char *name,struct _videoconfig vc)
    {*(PCT *)this=sign_storage_rgb(name,vc);
    if (!s_cols) return 1;
    PathName=MakeName(name);
    if (read_proto_SGN())
        {free_PCT();
        return 1;
        }
    return 0;
    }
//-----
int read_proto_DBC(FILE *datfp)
    {
        char now[80];
        if(fscanf(datfp,"%s ",now)==EOF)goto
ERR;PathName=MakeName(now);
        if(fscanf(datfp,"%s ",now)==EOF)goto
ERR;FRAME_Number=now;
        if(fscanf(datfp,"%s ",now)==EOF)goto
ERR;STRING_name=now;
        if(fscanf(datfp,"%s ",now)==EOF)goto ERR; SIGN_name=now;
        if(fscanf(datfp,"%d ",&(this->s_cols))==EOF)goto ERR;
        FRAME_Number.MakeUpper();
        STRING_name.MakeUpper();
        SIGN_name.MakeUpper();
        return 1;
    ERR: return 0;
    }

```

```

//=====
===
int alloc_Charact_dbl(long Num)
{
    if(!(Charact=new double[Num])) return 1;
    MaxNum=Num; NumberOfChk=0;
    return 0;
}
//-----
void free_Charact()
{delete Charact;
 Charact=NULL;
}
//-----
int read_Charact_dbl(FILE *inp,long Num)
{short i;
 double d;
 if(MaxNum<(NumberOfChk+Num)) return 1;
 for (i=0;i<Num;i++)
 {if(fscanf(inp,"%lf",&d) ==EOF) return 1;
  if(fabs(d)<1.0e-4) d=0;
   Charact[NumberOfChk]=d;
   NumberOfChk++;
 }
 return 0;
}
//-----
double CorrValue(short WNum,short Pnum)
{return (*(Charact+(long)WNum*s_cols+Pnum));
}
//-----
};
//=====

#define UnKnown -1
//=====
=
typedef struct
{ short n; // voiting numbers
  short pos; // position in string
  double value; //value
} RSLT;

//-----
void HistCollect(short NOfWin,short St,short Fin,PRT &Db);
RSLT LineEstimation (short TagSize, PRT &Db,short NOfWin,
                    short WSize,double Thr);
int LineInf(const PRT &P, PRT T, short rw, short Xpos,struct _videoconfig vc);
double LinInter( PRT &P,short WNum,short WSize ,double Pt);

```

212

SUBSTITUTE SHEET (RULE 26)

```
void HistThresh(short *H,short *BotThr,short *TopThr,short num);
```

```
#endif
```

al3

SUBSTITUTE SHEET (RULE 26)

```

#ifndef DECISION
#define DECISION
#include <iostream.h>
#include "PIC_PRO.h"
#define UnKnown -1
//=====
=
typedef struct
{ short n; // voiting numbers
  short pos; // position in string
  double value; //value
} RSLT;

//-----
void HistThresh(short *H,short *BotThr,short *TopThr,short num)

void HistCollect (short **His, short NOfWin, PRT &Db);
RSLT LineEstimation (short TagSize, PRT Db,short NOfWin,
                    short WSize,double Thr);
int WriteLineInf(PRT &P, PRT T, short rw, short Xpos);

```

```
#ifndef LNS_CORR
#define LNS_CORR
#include "projctn7.h"
#include "pic_mch7.h"
#include "res_mch7.h"
#include "lin_tm7.h"
// #include "tem_plt7.h"

#define NAME_LENGTH 40
#define GRAPHMODE _VRES256COLOR

#define HistDim 101

#endif
```

```

//          PRT_ANLS
//      Module analyses file PROTO_1.dbc (output of LNS_CORR),
//calculates histogramme functions of PROTO_1 in file PROTO_1.hs2.
//Bisides that it asks a possibilites to connect PROTO_1
//with another sign in one strip.
//
// COMMAND STRING
//
// prt_anls <PROTO_1_Name> [CommandFile]
//
//      <PROTO_1_Name>      File name of PROTOTYPE without
extention
//      [CommandFile]      Optional ASCII file with a run time parameters.
//
// INPUT
//      .DBC, .RGB files of prototypes and corresponding .SGN files created
by
// modules LNS_CORR and MAKEPRB.
// RUN TIME parameters:
//
//      <CORR_THR>      threshold for sign linking
//                      We have used 0.6;
//                      If cross correlation axceede the <CORR_THR> the number of
//                      string will be asked. Negative number cause not including the
//                      connection in list.
//SEE ALSO FILE "PRT_ANLS.INI"
// OUTPUT
//      PROTO_1.HS2 - Histogramme ;
//      LINECOL.PRT - File collects an information about sign linking
//                      in accordance with <CORR_THR> and our response.
//      FILE structure:
// (example file string start after char # that not belongs to file
// *****0 pos in str*****
//      #LineCollection
//      #Line #  FName #1  FName #2  Pos ...
//      #1      c:\ilyalwinps\s160_0    c:\ilyalwinps\s160_6      18
//      #1      c:\ilyalwinps\s160_0    c:\ilyalwinps\s160_15     37
//EOF
// Here 1 Line number named through the keyboard
//....s160_0 - PROTO_1 name;
//....s160_6 and s160_15 name  of prototypes linked with a PROTO_1
//      in one strip;
// 18 and 37 positions of corresponding prototypes start relative to
// 0 column of PROTO_1 (PROTO_1 STARTS in 0 column of PROTO_1)

#include <stdlib.h>
#include <conio.h>
#include <stdio.h>
#include <string.h>

```

```
#include <graph.h>
#include <float.H>
#include <time.h>
#include <ctype.h>
#include <fstream.h>
#include <afx.h>
#include "prt_anls.h"
#include "PIC_PRO.h"

#define MaxNumOfPicks 20
const short MaxNumOfWin =25;
#define MaxProtNum 25

short Hist[MaxNumOfWin][HistDim];

char LineColName[] ="linecol.prt";
char f_name[40]="_",FILE_name[40]="_", FRAME_Name[40]="_",
    STRING_name[40]="_", SIGN_name[40]="_";
double GAMMA=1.0,CORR_THRESH=0.0,Thresh_mdl=0.0;
short MAP;
short VOITING=3,TAG_hight;
struct _videoconfig vc;
double sh[4]={0,0,0,0};

int get_number_4(); // INITIALISATION GRAPHICMODE, GET SCALE
int get_number_4_f(FILE *f); // INITIALISATION GRAPHICMODE, GET
SCALE
//$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$
$$$$$$$$$$$$$$$$$$$$$

CString PROTOCOL_NAME;

ofstream LineC;
//===== OpenLineCollection
void OpenLineCol(const char*name)
{ LineC.open(name,ios::out|ios::app|ios::out|ios::nocreate);
if(LineC.fail())
{LineC.clear(0);
LineC.open(name,ios::out|ios::app|ios::out|ios::noreplace);
if(LineC.fail())
{LineC.clear(0);
cout << "CAN NOT OPEN LineCollection";
GRAPH_OUT(-1);
}
}
}
//=====
SCR_PNT winsize;
```

```

//=====
=====
int main(int argc,char* argv[])
{
int FLG_F=0;
FILE *datainf;
short n=0;
SCR_PNT t_pos;
if((argc != 2) && (argc !=3))
{
printf(" proto_file\n");
FLG_F=0;
return(1);
}
else
if(argc ==3)
{FLG_F=1;
if(!(datainf=fopen(argv[2],"r"))) return 0;
}
if(FLG_F)
get_number_4_f(datainf); // GET SCALE
else
get_number_4();

PRT prototype;
// ===== GRAPHICS START
if(GRAPHICS_START(&vc,GRAPHMODE)) GRAPH_OUT(-1);
//=====
if(prototype.proto_storage_rgb(argv[1],vc))
{printf("SGN TARGET not exist"); GRAPH_OUT(-1);
}
// READ .DBC FILE AND STORED Correlation Function
//===== PROTOTYPE LOOP OVER DBASE .dbc
PRT DbProto[MaxProtNum];
CString DbName=argv[1];
DbName+=" .dbc";
FILE *Db=fopen((const char *)DbName,"r");
if(!Db)
{printf("DBase not exist"); GRAPH_OUT(-1);}

// DEBUG Split two rows
short WinSize;
char AA [128],SPACE[14];
fscanf(Db,"%[^\\n]s ", AA);
fscanf(Db,"%[^\\n]s", AA);
fscanf(Db,"%*s %*s %*s %*s %*s %d %s",&WinSize,SPACE);

const MaxNumberOfWin=30;
short NumberOfWin=0;//=prototype.s_cols/WinSize;

```



```

short iDbProto=0;
fscanf(Db,"%[^\\n]s ", AA);
short i,j;
while( DbProto[iDbProto].read_proto_DBC(Db) )
{
    if(DbProto[iDbProto].alloc_Charact_dbl(MaxNumberOfWin*DbProto[iDbProto]
.s_cols) )
        {printf(" NOT MEMORY for data"); GRAPH_OUT(-1);}
    fscanf(Db," %[^\\n]s ",AA); // DEBUG Split one row
    while(!feof(Db))
    {
        NumberOfWin++;
        if(DbProto[iDbProto].read_Charact_dbl(Db,DbProto[iDbProto].s_cols))
            {printf(" END of DB"); GRAPH_OUT(-1);}
        fscanf(Db," %[^\\n]s ",AA); // DEBUG Split one row
    }
    iDbProto++;
}
fclose(Db);
// END READ .DBC FILE AND STORED Correlation Function
OpenLineCol(LineColName);
LineC<< "LineCollection \\nLine #\\t FName #1\\t FName #2\\t Pos ... \\n";
//LOOP over targets
int LN,p;
RSLT LineEst;
short St,Fin;
for (i=0;i<MaxNumOfWin;i++)
    for (j=0;j<HistDim;Hist[i][j++]=0);

//Miki's Threshold
short PrNumber=-1;
for(i=0;i<iDbProto;i++)
    if(DbProto[i].PathName==prototype.PathName)
        {PrNumber=i; break;}
// IF AutoCorr absence PrNumber=-1

// Line linking and noise calculation
for (i=0;i<iDbProto;i++)
{
    St=0;
    Fin= DbProto[i].s_cols;
    if(i!=PrNumber)
        {LineEst=LineEstimation(prototype.s_cols, DbProto[i],NumberOfWin,
                                WinSize,CORR_THRE
SH);
        if (LineEst.n)
        {
            p=LineEst.pos-DbProto[i].s_cols; // DbProtoStart

```

```

        LN=LineInf(prototype, DbProto[i], vc.numtextrows, p+10,vc);
        if (LN>=0)
        {if (p>0)
            St=prototype.s_cols-p;
            else
            Fin=-p;
            LineC<<LN<<"\t"<<prototype.PathName<<"\t"<<
            DbProto[i].PathName<<"\t"<<p<<"\n";
        }
    }
    //Histogramm calculation
    HistCollect(NumberOfWin,St,Fin,DbProto[i]);
}
LineC.close();
// RESULT OUT

PROTOCOL_NAME=argv[1];
PROTOCOL_NAME+="hs2";
OpenLineCol((const char*)PROTOCOL_NAME);
LineC<<"Histogrammes\n";
LineC<<argv[1]<<"\t"<<prototype.SIGN_name<<"\n";
LineC<<"NumberOfWindows\t"<<NumberOfWin<<"\n";
LineC<<"NumberOfBins\t"<<HistDim<<"\n";
LineC<<"Win_pos\n";

for(j=0;j<NumberOfWin;j++)
{LineC<<j*WinSize<<"\t";
  for(i=0;i<HistDim;i++)
    LineC<<Hist[j][i]<<"\t";
  LineC<<"\n";
}
LineC.close;
//CORRELATION PROCESSING
GRAPH_OUT();
return(0);
}

//
=====
=====
/*void get_shift_f(FILE *f,double * sh) // INITIALISATION GRAPHICMODE,
GET SCALE
{int i;
  for(i=0;i<4; i++)
  {
    fscanf(f,"%lf %lf\n",sh+i++,sh+i);
  }
}

```

a a a

```

//
=====
=====
void get_shift(double * sh) // INITIALISATION GRAPHICMODE, GET SCALE
{int i;
    cout<< "vertexes shift over rows ( top_right, bottom_right, bottom_left,
top_left %\n";
    for (i=0;i<4; i++)
        cin>>sh[i];
}*/
//=====
=====
int get_number_4() // INITIALISATION GRAPHICMODE, GET SCALE
{int R;
    GRAPH_OUT();
    cout << "CORR_THRESH \n";
    cin >>CORR_THRESH;
// ===== GRAPHICS START
    if(GRAPHICS_START(&vc,GRAPHMODE)) GRAPH_OUT(-1);
//=====
    return R;
}
//=====
int get_number_4_f(FILE *f) // INITIALISATION GRAPHICMODE, GET
SCALE
{
    fscanf(f," %lf",&CORR_THRESH);
    return 1;
}
//=====
/*int picture_inf_num_new(char *name,SCR_PNT *vertexes,short n)
{int i,j;
    char new_str[80];
    FILE *datfp;
    int r,FLG=0;
    strcat(strcpy(new_str,name),".sgn");
    if(!(datfp=fopen(new_str,"r"))) return 0;
    r=fscanf(datfp,"%[^\n]s",new_str);
    r=fscanf(datfp," %[^\n]s",new_str);
    if(__iscsymf( (int)new_str[0]))//FILE INFORMATION
        //(letter or underscore)
        {sscanf(new_str," %s %s %s %s",&FILE_name, &FRAME_Name,
&STRING_name, &SIGN_name);
        r=fscanf(datfp," %[^\n]s",new_str);
        }
    for(j=0;j<n+1;j++)
        for(i=0;i<4;i++)
            {if(FLG)
                if(fscanf(datfp," %[^\n]s",new_str)==EOF)
                    a2i

```

```
        {fclose (datfp); return 0;}
    FLG=1;
    sscanf(new_str," %d %d",&(vertexes[i].c),&(vertexes[i].r));
    }
    fclose(datfp);
    return 1;
} */
//=====
```

```

#include <io.h>
#include <iostream.h>
#include <fstream.h>
#include "prt_anls.h"
#include "PIC_PRO.h"

extern const short MaxHistSize;
extern const short MaxNumOfWin;

extern short Hist[][HistDim];
#define Histogramm(i,x) (Hist[(i)][(short)( 0.5*((x)+1)*(HistDim-1)+0.5)]++)
//=====
===
void HistCollect( short NOFWin,short St,short Fin,PRT &Db)
{short i,j;
 double val;
 for (i=0;i<NOFWin;i++)
   for(j=St;j<Fin;j++)
   {
     val=Db.CorrValue(i,j);
     Histogramm(i,val);
   }
}
//=====
=
RSLT LineEstimation(short TagSize, PRT &Db,short NOFWin,short
WSize,double Thr)
{RSLT out={0,-1,0.0};
 short i,j,EndPosDb=0,k;
 double *DMF=new double[k=Db.s_cols+TagSize];
 for (i=0;i<k;DMF[i++]=0.0);
 double val;
 for (i=0;i<NOFWin;i++)
   for(j=0;j<Db.s_cols;j++)
   {
     val=Db.CorrValue(i,j);
     if (val>Thr)
       {EndPosDb=i*WSize-j+Db.s_cols;
        DMF[EndPosDb]+=val;
        out.n++;
       }
   }
 if(out.n)
   for(i=0;i<k;i++)
     if(out.value<DMF[i])
       {out.value=DMF[i];
        out.pos=i;
       }
 delete DMF;

```

```

return out;
}
//=====
=====
int LineInf(const PRT &P, PRT T, short rw, short Xpos, struct _videoconfig vc)
{SCR_PNT PR(10,10),TG(0,18);
char mess[80]="LineNum [<0] UnKnown";
int out=UnKnown;
    TG.c=Xpos;
    _settextposition(rw,1 );
    *((PCT *)&T)=
    sign_storage_rgb((const char *)(T.PathName),vc);
    if (T.s_cols)
        { _setcolor( color_num(0,0,0));
          _rectangle( _GFillInterior,0,0, 680,60 );
          sign_present_RGB(P,PR);
          sign_present_RGB(T,TG);
          T.free_PCT();
          _setcolor( color_num(240,240,240));
          _outtext (mess);
          cin>>out;
        }
    else
        {
          sprintf(mess,"%s from %s RGB UNAGCESIBLE ",
            (const char *)(P.SIGN_name),
            (const char *)(T.STRING_name));
          _outtext (mess);
        }
return out;
}
//=====
=====
double CorVal( PRT &P,short WNum,short y)
{double v1;
if(y<0) v1=P.CorrValue(WNum,0);
else {if(y>=P.s_cols) v1=P.CorrValue(WNum,P.s_cols-1);
      else v1=P.CorrValue(WNum,(short)y);
    }
return v1;
}
//=====
=====
#define VALUE(X) ((X)<(P.s_cols-WSize))?CorVal(P, WNum,
(X)):CorVal(P, WNum, (X-1))
//=====
=====
double LinInter( PRT &P,short WNum,short WSize ,double Pt)
{double y1,y0,x=WNum*WSize+Pt;

```

```

    short x1,x0;
    x0=(short)x;
    x1=(x0>x)?x0-1:x0+1;
    y1=CorVal(P, WNum, x1);
    y0=CorVal( P, WNum, x0);
    return( y0+(x-x0)*(y1-y0)/(x1-x0));
}
//=====
void HistThresh(short *H,short *BotThr,short *TopThr,short num)
{short S=0;
 *BotThr=0;
 *TopThr=HistDim-1;
 while((S+=H[(*TopThr)--])<num);
 S=0;
 while((S+=H[(*BotThr)++])<num);
}
//=====
CString MakeName(CString N)
{
 short k=(N.SpanIncluding(" \t")).GetLength();
 char *p,fp[80];
 p=((char*)(const char *)N)+k;
 CString M=p;
 if(M.Find('.')<0)
 {if(M.GetAt(0)=='\')
 { M='.'+M;
 M= (char)(_getdrive( )-1+'A')+M;
 }
 else
 M=_fullpath(fp,(const char *)M,80);
 }
 M.MakeLower();
 return M;
}
//=====
=
CString MakeName(char *p )
{CString M(p);
 return (MakeName(M));
}
//=====
=====

```

ORIGIN = PWB  
ORIGIN\_VER = 2.1.49  
PROJ = TRACK  
PROJFILE = TRACK.MAK  
DEBUG = 1

NMAKEBSC1 = set  
NMAKEBSC2 = nmake  
CC = cl  
CFLAGS\_G = /AL /W2 /G2 /GA /DMSC /DMFG\_ /DWIN /DPAS /DDLL /GEf  
/Zp /BATCH\  
/FR\$\*.sbr  
CFLAGS\_D = /f- /Od /FPI87 /Zi /Gs  
CFLAGS\_R = /f- /Os /Og /Oe /FPI87 /Gs  
CXX = cl  
CXXFLAGS\_G = /G2 /W2 /GA /GEf /Zp /BATCH /FR\$\*.sbr  
CXXFLAGS\_D = /f /Zi /Od /Gs  
CXXFLAGS\_R = /f- /Oe /Og /Os /Gs  
MAPFILE\_D = NUL  
MAPFILE\_R = NUL  
LFLAGS\_G = /STACK:16000 /BATCH /ONERROR:NOEXE  
LFLAGS\_D = /CO /NOF  
LFLAGS\_R = /NOF  
LLIBS\_G = c:\c700\windev\lib\LIBW.LIB c:\c700\windev\lib\commdlg.lib\  
c:\visnplus\lib\win\_ai.lib c:\visnplus\lib\mfghmm.lib\  
c:\visnplus\lib\liffdll.lib  
LINKER = link  
ILINK = ilink  
LRF = echo > NUL  
ILFLAGS = /a /e  
RC = rc -K  
LLIBS\_R = /NOD:LLIBC7 LLIBC7W  
LLIBS\_D = LLIBCEW /NOD:LLIBC7  
BRFLAGS = /o \$(PROJ).bsc  
BSCMAKE = bscmake  
SBRPACK = sbrpack  
BROWSE = 1  
PACK\_SBRs = 1



227

```

FILES = BITMAP.C MIN_MAG.C PERSP.C TRACK.C TRACK.DEF
TRACK.RC LIB.C LINES.C\
    QRSOLV.C
DEF_FILE = TRACK.DEF
OBJS = BITMAP.obj MIN_MAG.obj PERSP.obj TRACK.obj LIB.obj
LINES.obj\
    QRSOLV.obj
RESS = TRACK.res
SBRS = BITMAP.sbr MIN_MAG.sbr PERSP.sbr TRACK.sbr LIB.sbr
LINES.sbr\
    QRSOLV.sbr

```

```
all: $(PROJ).exe
```

```
.SUFFIXES:
```

```
.SUFFIXES:
```

```
.SUFFIXES: .obj .sbr .res .c .rc
```

```
BITMAP.obj : BITMAP.C const.h bitmap.h lines.h track.h min_mag.h lib.h
!IF $(DEBUG)
```

```
    @$(CC) @<<$(PROJ).rsp
```

```
/c $(CFLAGS_G)
```

```
$(CFLAGS_D) /FoBITMAP.obj BITMAP.C
```

```
<<
```

```
!ELSE
```

```
    @$(CC) @<<$(PROJ).rsp
```

```
/c $(CFLAGS_G)
```

```
$(CFLAGS_R) /FoBITMAP.obj BITMAP.C
```

```
<<
```

```
!ENDIF
```

```
BITMAP.sbr : BITMAP.C const.h bitmap.h lines.h track.h min_mag.h lib.h
```

```
!IF $(DEBUG)
```

```
    @$(CC) @<<$(PROJ).rsp
```

```
/Zs $(CFLAGS_G)
```

```
$(CFLAGS_D) /FRBITMAP.sbr BITMAP.C
```

```
<<
```

227

!ELSE

    @\$(CC) @<<\$(PROJ).rsp

/Zs \$(CFLAGS\_G)

\$(CFLAGS\_R) /FRBITMAP.sbr BITMAP.C

<<

!ENDIF

MIN\_MAG.obj : MIN\_MAG.C const.h bitmap.h lines.h track.h persp.h  
min\_mag.h\

lib.h

!IF \$(DEBUG)

    @\$(CC) @<<\$(PROJ).rsp

/c \$(CFLAGS\_G)

\$(CFLAGS\_D) /FoMIN\_MAG.obj MIN\_MAG.C

<<

!ELSE

    @\$(CC) @<<\$(PROJ).rsp

/c \$(CFLAGS\_G)

\$(CFLAGS\_R) /FoMIN\_MAG.obj MIN\_MAG.C

<<

!ENDIF

MIN\_MAG.sbr : MIN\_MAG.C const.h bitmap.h lines.h track.h persp.h  
min\_mag.h\

lib.h

!IF \$(DEBUG)

    @\$(CC) @<<\$(PROJ).rsp

/Zs \$(CFLAGS\_G)

\$(CFLAGS\_D) /FRMIN\_MAG.sbr MIN\_MAG.C

<<

!ELSE

    @\$(CC) @<<\$(PROJ).rsp

/Zs \$(CFLAGS\_G)

\$(CFLAGS\_R) /FRMIN\_MAG.sbr MIN\_MAG.C

<<

!ENDIF

```

PERSP.obj : PERSP.C const.h bitmap.h lines.h track.h persp.h min_mag.h
lib.h
!IF $(DEBUG)
    @$(CC) @<<$(PROJ).rsp
/c $(CFLAGS_G)
$(CFLAGS_D) /FoPERSP.obj PERSP.C
<<
!ELSE
    @$(CC) @<<$(PROJ).rsp
/c $(CFLAGS_G)
$(CFLAGS_R) /FoPERSP.obj PERSP.C
<<
!ENDIF

```

```

PERSP.sbr : PERSP.C const.h bitmap.h lines.h track.h persp.h min_mag.h
lib.h
!IF $(DEBUG)
    @$(CC) @<<$(PROJ).rsp
/Zs $(CFLAGS_G)
$(CFLAGS_D) /FRPERSP.sbr PERSP.C
<<
!ELSE
    @$(CC) @<<$(PROJ).rsp
/Zs $(CFLAGS_G)
$(CFLAGS_R) /FRPERSP.sbr PERSP.C
<<
!ENDIF

```

```

TRACK.obj : TRACK.C const.h bitmap.h persp.h lines.h track.h min_mag.h
lib.h
!IF $(DEBUG)
    @$(CC) @<<$(PROJ).rsp
/c $(CFLAGS_G)
$(CFLAGS_D) /FoTRACK.obj TRACK.C
<<
!ELSE
    @$(CC) @<<$(PROJ).rsp
/c $(CFLAGS_G)

```

```
$(CFLAGS_R) /FoTRACK.obj TRACK.C
<<
!ENDIF
```

```
TRACK.sbr : TRACK.C const.h bitmap.h persp.h lines.h track.h min_mag.h
lib.h
!IF $(DEBUG)
    @$(CC) @<<$(PROJ).rsp
    /Zs $(CFLAGS_G)
    $(CFLAGS_D) /FRTRACK.sbr TRACK.C
    <<
!ELSE
    @$(CC) @<<$(PROJ).rsp
    /Zs $(CFLAGS_G)
    $(CFLAGS_R) /FRTRACK.sbr TRACK.C
    <<
!ENDIF
```

```
TRACK.res : TRACK.RC track.h frames.dlg
    $(RC) $(RCFLAGS1) /r /fo TRACK.res TRACK.RC
```

```
LIB.obj : LIB.C const.h bitmap.h persp.h lines.h track.h min_mag.h lib.h
!IF $(DEBUG)
    @$(CC) @<<$(PROJ).rsp
    /c $(CFLAGS_G)
    $(CFLAGS_D) /FoLIB.obj LIB.C
    <<
!ELSE
    @$(CC) @<<$(PROJ).rsp
    /c $(CFLAGS_G)
    $(CFLAGS_R) /FoLIB.obj LIB.C
    <<
!ENDIF
```

```
LIB.sbr : LIB.C const.h bitmap.h persp.h lines.h track.h min_mag.h lib.h
!IF $(DEBUG)
    @$(CC) @<<$(PROJ).rsp
    /Zs $(CFLAGS_G)
```

```
$(CFLAGS_D) /FRLIB.sbr LIB.C
```

```
<<
```

```
!ELSE
```

```
    @$(CC) @<<$(PROJ).rsp
```

```
/Zs $(CFLAGS_G)
```

```
$(CFLAGS_R) /FRLIB.sbr LIB.C
```

```
<<
```

```
!ENDIF
```

```
LINES.obj : LINES.C const.h bitmap.h persp.h lines.h track.h min_mag.h lib.h
```

```
!IF $(DEBUG)
```

```
    @$(CC) @<<$(PROJ).rsp
```

```
/c $(CFLAGS_G)
```

```
$(CFLAGS_D) /FoLINES.obj LINES.C
```

```
<<
```

```
!ELSE
```

```
    @$(CC) @<<$(PROJ).rsp
```

```
/c $(CFLAGS_G)
```

```
$(CFLAGS_R) /FoLINES.obj LINES.C
```

```
<<
```

```
!ENDIF
```

```
LINES.sbr : LINES.C const.h bitmap.h persp.h lines.h track.h min_mag.h lib.h
```

```
!IF $(DEBUG)
```

```
    @$(CC) @<<$(PROJ).rsp
```

```
/Zs $(CFLAGS_G)
```

```
$(CFLAGS_D) /FRLINES.sbr LINES.C
```

```
<<
```

```
!ELSE
```

```
    @$(CC) @<<$(PROJ).rsp
```

```
/Zs $(CFLAGS_G)
```

```
$(CFLAGS_R) /FRLINES.sbr LINES.C
```

```
<<
```

```
!ENDIF
```

```
QRSOLV.obj : QRSOLV.C qrsolv.h
```

```
!IF $(DEBUG)
```

```
    @$(CC) @<<$(PROJ).rsp
```

```

/c $(CFLAGS_G)
$(CFLAGS_D) /FoQRSOLV.obj QRSOLV.C
<<
!ELSE
    @$(CC) @<<$(PROJ).rsp
/c $(CFLAGS_G)
$(CFLAGS_R) /FoQRSOLV.obj QRSOLV.C
<<
!ENDIF

```

```

QRSOLV.sbr : QRSOLV.C qrsolv.h
!IF $(DEBUG)
    @$(CC) @<<$(PROJ).rsp
/Zs $(CFLAGS_G)
$(CFLAGS_D) /FRQRSOLV.sbr QRSOLV.C
<<
!ELSE
    @$(CC) @<<$(PROJ).rsp
/Zs $(CFLAGS_G)
$(CFLAGS_R) /FRQRSOLV.sbr QRSOLV.C
<<
!ENDIF

```

```

$(PROJ).bsc : $(SBRS)
    $(BSCMAKE) @<<
$(BRFLAGS) $(SBRS)
<<

```

```

$(PROJ).exe : $(DEF_FILE) $(OBJS) $(RESS)
    -$(NMAKEBSC1) MAKEFLAGS=
    -$(NMAKEBSC2) $(NMFLAGS) -f $(PROJFILE) $(PROJ).bsc
!IF $(DEBUG)
    $(LRF) @<<$(PROJ).lrf
$(RT_OBJS: = +^
) $(OBJS: = +^
)
$@

```

232

```

$(MAPFILE_D)
$(LIBS: = +^
) +
$(LLIBS_G: = +^
) +
$(LLIBS_D: = +^
)
$(DEF_FILE) $(LFLAGS_G) $(LFLAGS_D);
<<
!ELSE
    $(LRF) @<<$(PROJ).lrf
$(RT_OBJS: = +^
) $(OBJS: = +^
)
$@
$(MAPFILE_R)
$(LIBS: = +^
) +
$(LLIBS_G: = +^
) +
$(LLIBS_R: = +^
)
$(DEF_FILE) $(LFLAGS_G) $(LFLAGS_R);
<<
!ENDIF
    $(LINKER) @$$(PROJ).lrf
    $(RC) $(RCFLAGS2) $(RESS) $@

.c.obj:
!!IF $(DEBUG)
    @$$(CC) @<<$(PROJ).rsp
/c $(CFLAGS_G)
$(CFLAGS_D) /Fo$@ $<
<<
!ELSE
    @$$(CC) @<<$(PROJ).rsp
/c $(CFLAGS_G)

```

```
$(CFLAGS_R) /Fo$@ $<
```

```
<<
```

```
!ENDIF
```

```
.c.sbr :
```

```
!IF $(DEBUG)
```

```
    @$(CC) @<<$(PROJ).rsp
```

```
/Zs $(CFLAGS_G)
```

```
$(CFLAGS_D) /FR$@ $<
```

```
<<
```

```
!ELSE
```

```
    @$(CC) @<<$(PROJ).rsp
```

```
/Zs $(CFLAGS_G)
```

```
$(CFLAGS_R) /FR$@ $<
```

```
<<
```

```
!ENDIF
```

```
.rc.res :
```

```
$(RC) $(RCFLAGS1) /r /fo $@ $<
```

```
run: $(PROJ).exe
```

```
    WX $(WXFLAGS) $(PROJ).exe $(RUNFLAGS)
```

```
debug: $(PROJ).exe
```

```
    WX /p $(WXFLAGS) CVW $(CVFLAGS) $(PROJ).exe $(RUNFLAGS)
```



NAME      TRANSFORM

DESCRIPTION    'Changing signs'

EXETYPE       WINDOWS

STUB          'WINSTUB.EXE'

CODE        PRELOAD MOVEABLE DISCARDABLE

DATA        PRELOAD MOVEABLE MULTIPLE

HEAPSIZE     1024

STACKSIZE   8192

```
#include <windows.h>
#include "track.h"
#include "frames.dlg"
```

```
trans MENU
```

```
{
  POPUP "File"
  {
    MENUITEM "Open...",      IDM_OPEN
    MENUITEM "Write...",     IDM_WRITE
  }
}
```

```
POPUP "&Pick sign"
```

```
{
  MENUITEM "&Original",      IDM_PICK_ORIG
  MENUITEM "&Substitue",     IDM_PICK_SUBST
  MENUITEM "&Load Orig",     IDM_LOAD_ORIG
  MENUITEM "&Pick Corners",  IDM_PICK_CORNERS
}
MENUITEM "&change sign",    IDM_CHANGE
```

```
POPUP "&Method"
```

```
{
  MENUITEM "&Bilinear",      IDM_BILINEAR
  MENUITEM "&Trilinear",     IDM_TRILINEAR
  MENUITEM "&SplitMode",     IDM_SPLIT
}
MENUITEM "&Info",           IDM_INFO
```

```
POPUP "&MFG"
```

```
{
  MENUITEM "&Init"      DISP_INIT
  MENUITEM "&Grab"      DISP_GRAB
  MENUITEM "&Snap"      DISP_SNAP
  MENUITEM "&Load Pict"  DISP_LOAD
  MENUITEM "&Load Field" DISP_LOAD_FIELD
  MENUITEM "&Draw"      DISP_DRAW
}
```

236

```
MENUITEM "&Clean"  DISP_CLEAN
MENUITEM "&Wipe"   DISP_WIPE
}
MENUITEM "&Start track", IDM_START_TRACK
MENUITEM "&Tracking",      IDM_AUTO_TRACK

POPUP "&V-Disk"
{
  MENUITEM "&Init", SONY_INIT
  MENUITEM "&Go to", SONY_FRAME
  MENUITEM "&Replace", SONY_RECORD
  MENUITEM "&Close", SONY_END
}
}
```

```
frames DIALOG 19, 22, 126, 57
STYLE DS_MODALFRAME | WS_POPUP | WS_VISIBLE | WS_CAPTION |
WS_SYSMENU
CAPTION "Edit Frames"
FONT 8, "MS Sans Serif"
BEGIN
    LTEXT      "From Frame", 101, 10, 4, 48, 8
    LTEXT      "To Frame", 102, 10, 22, 56, 8
    EDITTEXT   103, 67, 4, 45, 13, ES_AUTOHSCROLL
    EDITTEXT   104, 67, 21, 45, 14, ES_AUTOHSCROLL
    PUSHBUTTON "OK", 105, 7, 39, 40, 14
    PUSHBUTTON "Cancel", 106, 72, 38, 40, 14
END
```

```

long FAR PASCAL _export WndProc (HWND, UINT, UINT, LONG);
DWORD GetDibInfoHeaderSize (BYTE huge *);
int PASCAL GetDibWidth (BYTE huge *);
int PASCAL GetDibHeight (BYTE huge *);
BYTE huge * GetDibBitsAddr (BYTE huge *);
DWORD GetDrawTableSize (BYTE huge *);
BYTE huge * ReadDib (HWND, char *, int*, int*);
int PASCAL display_information (HWND);
int PASCAL teach_grey_pallette (HWND, int);
BYTE huge * ReadRGB (HWND, char *, int*, int*);
int PASCAL draw_rgb_picture (HWND, HDC);
int PASCAL teach_rgb_pallette (HWND);
int PASCAL get_file_type_by_name (char *);
int PASCAL create_poly_src_dst();
int PASCAL keep_subst (HWND, MYBITMAP*);
BYTE PASCAL get_palette_index (BYTE, BYTE, BYTE);
int PASCAL WriteGreyRgb (HWND hwnd, char *szFileName, MYBITMAP
*Bmap);
int PASCAL get_in_series_flag();
int PASCAL smooth_values (double, double *, double *, int, double *);
int PASCAL enlarge_area_of_noise (MYBITMAP *, MYBITMAP *);

```

```

#define GREY_MODEL 1
#define COLOR_MODEL 2
#define RGBA_MODEL 3

```

```

#define IN_BILINEAR 2
#define IN_TRILINEAR 3

```

```

#define FILE_IS_BMP 1
#define FILE_IS_RGB 2
#define FILE_IS_UNKNOWN 3

```

```

#define AF_ZOOM 102
#define AF_ANGLE 104
#define AF_OK 105
#define AF_CANCEL 106

```

```
#define PER_WIDTH      103
#define PER_HEIGHT     104
#define PER_FROM_TOP   107
#define PER_FROM_BOTTOM 108
#define PER_OK         109
#define PER_CANCEL     110
```

```
#define IDM_OPEN 100
#define IDM_PICK 101
#define IDM_CHANGE 104
#define IDM_PERSPECTIVE 102
#define IDM_INFO 103
#define IDM_WRITE 105
#define IDM_PICK_ORIG 106
#define IDM_PICK_SUBST 107
#define IDM_LOAD_ORIG 108
#define IDM_PICK_CORNERS 109
```

```
#define IDM_NEAREST 110
#define IDM_BILINEAR 111
#define IDM_TRILINEAR 112
#define IDM_SPLIT 114
```

```
#define IDM_START_TRACK 150
#define IDM_AUTO_TRACK 151
```

```
#define DISP_SNAP 200
#define DISP_GRAB 201
#define DISP_LOAD 202
#define DISP_DRAW 203
#define DISP_CLEAN 204
#define DISP_WIPE 205
#define DISP_INIT 206
#define DISP_LOAD_FIELD 207
```

```
#define IDM_DEBUG 300
```

```
#define SONY_HEADER 400
#define SONY_FRAME 401
#define SONY_RECORD 402
#define SONY_INIT 403
#define SONY_END 404
```

```
#define SONY_FROM_FR 103
#define SONY_TO_FR 104
#define SONY_OK 105
#define SONY_CANCEL 106
```

```
typedef struct phdr {
    int cols, rows ;
    int bp ;
} PHDR ;
```

```
typedef BYTE TRIPLE[3] ;
```

```
typedef struct edge {
    double xs, ys;
    double xe, ye;
    int vertical;
    LINE l;
} EDGE;
```

```
typedef struct shift {
    double dx, dy;
    double sim; /* minimal value of sum of abs. diff. (MAD)
                maximal value of correlation (COR) */
} SHIFT;
```

```
typedef struct sign {
    MYBITMAP *Bmap ;
    MYBITMAP *BmapGrey ;
    TRACK_POINTS TrPoints ;
} SIGN ;
```

```
#include<math.h>
#include<stdio.h>
```

```
typedef unsigned char byte;
```

```
typedef struct {
    unsigned long
        r : 8,
        g : 8,
        b : 8,
        a : 8;
} RGBA;
```

```
typedef struct {
    double x ;
    double y ;
} RPOINT ;
```

```
typedef struct {
    byte r, g, b;
} MYRGB;
```

```
typedef enum {
    Grey, Rgb, Rgba
} BMTYPE;
```

```
typedef struct {
    int cols, rows;
    BMTYPE typ;
    BYTE huge *gpic;
} MYBITMAP;
```

```
typedef double LUMARR[7][7];
```

```
#define BITMAP_PLACE(bm,i,j) \
    *((BYTE huge*)((bm->gpic)+(DWORD)(i)*(DWORD)(bm->cols)+(DWORD)(j)))
#define BITMAP_PLACE_PTR(bm,i,j) \
```



```

        (BYTE huge*)((bm->gpic)+(DWORD)(i)*(DWORD)(bm->cols)+(DWORD)(j))

#define BITMAP_RGB_PLACE(bm,i,j) \
    *((BYTE huge*)((bm->gpic)+((DWORD)(i)*(DWORD)(bm->cols)+(DWORD)(j))*(DWORD)3))
#define BITMAP_RGB_PLACE_PTR(bm,i,j) \
    (BYTE huge*)((bm->gpic)+ ((DWORD)(i)*(DWORD)(bm->cols)+(DWORD)(j))*(DWORD)3)

#define MIN(a, b) ((a) < (b) ? (a) : (b))
#define MAX(a, b) ((a) > (b) ? (a) : (b))

#define DLVLS 3
#define ILVLS 1

#define MAX_NO_OF_TR_WIN 30

typedef struct {
    double x, y;
} PNT;

typedef struct {
    double    Af[3][3];
    double    Ab[3][3];
}    Affine_Transform;

typedef struct {
    double    Pf[3][3];
    double    Pb[3][3];
}    Perspective_Transform;

typedef struct track_points {
    POINT TrackP[NUM_OF_TRACK_POINTS];
    int NumOfPoints;
} TRACK_POINTS;

typedef struct shift_points {
    RPOINT TrackP[NUM_OF_TRACK_POINTS];

```

```

    int NumOfPoints ;
} SHIFT_POINTS ;

```

```

typedef struct tr_win {
    RPOINT Center ;
    double Xdir ;
    double Ydir ;
} TR_WIN ;

```

```

MYBITMAP FAR*  bm_alloc(int , int , BMTYPE ) ;
double PASCAL bilinear(double , double , MYBITMAP FAR *) ;
double PASCAL bilinear_red(double , double , MYBITMAP FAR *) ;
double PASCAL bilinear_green(double , double , MYBITMAP FAR *) ;
double PASCAL bilinear_blue(double , double , MYBITMAP FAR *) ;
int PASCAL multmat(HWND,int , int , int , double *, double *, double *) ;
double PASCAL resample_trilinear(MYBITMAP FAR*
,double,double,double,int) ;
double PASCAL resample_trilinear_red(MYBITMAP FAR*
,double,double,double,int) ;
double PASCAL resample_trilinear_blue(MYBITMAP FAR*
,double,double,double,int) ;
double PASCAL resample_trilinear_green(MYBITMAP FAR*
,double,double,double,int) ;
int PASCAL build_pyramid(HWND,MYBITMAP FAR *) ;
int PASCAL bm_free(MYBITMAP FAR* ) ;
int PASCAL create_lum_bmap(MYBITMAP *, MYBITMAP **);
int PASCAL duplicate_bmap(MYBITMAP *, MYBITMAP **,int ) ;
int PASCAL create_grey_bounded_bitmap(RECT, MYBITMAP *,MYBITMAP
**);
int PASCAL subtract_bitmaps(RECT,MYBITMAP *,MYBITMAP *,MYBITMAP
**);
int PASCAL insert_grey_bounded_bitmap(RECT ,MYBITMAP *, MYBITMAP
*) ;
int PASCAL get_mask_bitmap(HWND,MYBITMAP *,int ,MYBITMAP **);
int PASCAL find_horiz_line(MYBITMAP*,POINT , POINT , RPOINT *) ;
int PASCAL find_vertic_line(MYBITMAP *,POINT , POINT , RPOINT *) ;
double PASCAL bilinear_rgb(double, double , MYBITMAP FAR*,double *) ;

```

```
int PASCAL copy_grey_rect_from_frame(MYBITMAP *, MYBITMAP *,RECT )  
;  
int PASCAL build_alpha_map(MYBITMAP **, MYBITMAP*,RPOINT *) ;  
int PASCAL split_bitmap(MYBITMAP *,int,int ) ;  
int PASCAL split_bitmap_frame(MYBITMAP *,MYBITMAP **, MYBITMAP **) ;  
int PASCAL sum_grey_bitmap_value(MYBITMAP *,DWORD *) ;  
int PASCAL filter_noises_by_rects(MYBITMAP *,RECT ,int , int ,MYBITMAP  
*) ;
```

```

#define VALUE_IN(array,i,j,cols) *(array +(DWORD)(i)*(DWORD)(cols) +(j))
#define X_DIM 1
#define Y_DIM 2

int PASCAL l_cp_int_arr_to_RPOINT(RPOINT *, int *, int *,int );
int PASCAL l_quad_in_new_origin(RPOINT *,RPOINT *,int ,int ,int );
int PASCAL find_extremes_in_1dim(RPOINT *, int , int ,double *, double *);
int PASCAL find_best_cluster(HWND, int,RPOINT *,RPOINT *,int
*,int,int*,int*,
                                RPOINT*,int*);
int PASCAL l_copy_RPOINT_array(RPOINT *,RPOINT *,int );
int PASCAL l_copy_int_array(int *,int *,int );
int PASCAL l_find_bound_rect(RPOINT *, RECT *);
int PASCAL print_transform(HFILE ,char *,Perspective_Transform *);
int PASCAL insert_new_vertexes(RPOINT *CurrVert, RPOINT *Vert1,
RPOINT *Vert2,
                                RPOINT *Vert3, RPOINT *Vert4,double *Wheight,int num,RPOINT
*NewVert);
int PASCAL transform_rpoint_arr(RPOINT *,RPOINT
*,int,Perspective_Transform);

```

```

#define FAC 16
#define LOBES 2
#define FSIZE (LOBES * FAC)
#define CORR_WINDOWX 8
#define CORR_WINDOWY 6

MYBITMAP FAR * minify(HWND, MYBITMAP FAR *, int);
int PASCAL sinc_filter(int);
MYBITMAP FAR * hminify(HWND, MYBITMAP FAR *, int);
MYBITMAP FAR * vminify(HWND, MYBITMAP FAR *, int);
int PASCAL lpf1D(BYTE huge*, int, int, BYTE huge*);
int PASCAL lpf1D_rgb(BYTE huge*, int, int, BYTE huge*);
int PASCAL edge_refine(MYBITMAP *, EDGE *, int);
int PASCAL h_refine(MYBITMAP *, double *, double *, int);
int PASCAL v_refine(MYBITMAP *, double *, double *, int);
int PASCAL sub_pixel_interp(double, double, double, double *, double *em);
int PASCAL xysolve(HWND, MYBITMAP *, MYBITMAP *, SHIFT *, TR_WIN *,
int,
    TRACK_POINTS*,
    TRACK_POINTS*, Perspective_Transform*, HFILE,
    TRACK_POINTS*, RPOINT*);
int PASCAL xysrch(HWND, MYBITMAP *, MYBITMAP *, SHIFT
*, POINT, POINT,
int, int, int, int);

int PASCAL sub_pixel_refine(int, int, int, int, SHIFT *);
int PASCAL find_average_shift(HWND, int, SHIFT
*, SHIFT_POINTS, SHIFT_POINTS,
    Perspective_Transform*);

int PASCAL Quad2Quad(HWND, RPOINT srcpnts[4], RPOINT dstpnts[4],
    Perspective_Transform *);

int PASCAL copy_transform(Perspective_Transform *, Perspective_Transform
*);
int PASCAL Thin_Perspective(HWND, SHIFT_POINTS, SHIFT_POINTS
, int,

```

```
Perspective_Transform **);  
int PASCAL trans_grey_frame_to_fields(HWND ,RPOINT *,RPOINT *,  
MYBITMAP *,RPOINT *, MYBITMAP **,MYBITMAP **);
```

```

#define det2(a11, a12, a21, a22) (a11 * a22 - a12 * a21)

int PASCAL perspective(HWND, MYBITMAP FAR *, MYBITMAP
FAR *, RPOINT *, RPOINT *, int,
    Perspective_Transform * );
int PASCAL Rectan2Quad(HWND, RPOINT *, RPOINT
*, Perspective_Transform * );
int PASCAL Perspective_map(MYBITMAP FAR *, Perspective_Transform *,
    MYBITMAP FAR *, RPOINT *, int );
RPOINT bPerspective(RPOINT, Perspective_Transform * );
RPOINT PASCAL dPerspective(RPOINT, Perspective_Transform * );
int PASCAL check_if_rect(RPOINT * );
RPOINT PASCAL fPerspective(RPOINT, Perspective_Transform * );

int PASCAL median_filter_5(HWND, MYBITMAP * );
int PASCAL get_tresh_for_occ(MYBITMAP *, int * );
int PASCAL perspective_mask(HWND, MYBITMAP *, MYBITMAP *,
MYBITMAP *,
    RPOINT *, RPOINT *, int, int, MYBITMAP * );
int PASCAL Perspective_map_mask(MYBITMAP FAR
*, Perspective_Transform *,
    MYBITMAP FAR *, MYBITMAP *, RPOINT *, int, int, MYBITMAP * );
int PASCAL Quad2Rectan(HWND, RPOINT *, RPOINT
*, Perspective_Transform * );
int PASCAL perspective_al(HWND, MYBITMAP FAR *, MYBITMAP FAR
*, MYBITMAP *,
    RPOINT *, RPOINT *, int, Perspective_Transform * );
int PASCAL Perspective_map_al(MYBITMAP *, Perspective_Transform *,
MYBITMAP *, MYBITMAP *, RPOINT *, int );
int PASCAL Perspective_near_map(MYBITMAP
*, Perspective_Transfor
m *,
    MYBITMAP *, RPOINT *, int );
int PASCAL perspective_near(HWND, MYBITMAP *, MYBITMAP *,
    RPOINT *, RPOINT *, int, Perspective_Transform * );

```

```

#include <windows.h>
#include <windowsx.h>
#include <commdlg.h>
#include <stdlib.h>
#include <bios.h>
#include "const.h"
#include "bitmap.h"
#include "persp.h"
#include "lines.h"
#include "track.h"
#include "min_mag.h"
#include "lib.h"
#undef RGB
#include <mfgghost.h>

```

```

#define UNTIL_FRAME 420
#define UNTIL_PICT 4

```

```

int PASCAL bitmap_for_display(HWND,HDC,MYBITMAP FAR *,RECT);
int PASCAL draw_to_screen(MYBITMAP *,int,int);
int PASCAL create_disp_bmap(MYBITMAP *, MYBITMAP **,RECT);
int PASCAL pick_sign(HWND , UINT , UINT , LONG );
int PASCAL change_sign(HWND , UINT , UINT , LONG );
int PASCAL add_tracking_point(HWND , UINT , UINT , LONG );
int PASCAL change_sign_by_tracking(HWND,int);
int PASCAL WriteRgb (HWND ,char *,MYBITMAP *);
int PASCAL create_norm_bmap(HWND ,MYBITMAP *,MYBITMAP**,RECT);
int PASCAL keep_orig(HWND ,MYBITMAP *);
int PASCAL pick_original(HWND, UINT, UINT, LONG);
int PASCAL create_mask_bmap(HWND ,MYBITMAP *,RPOINT *,RPOINT
*,RECT,MYBITMAP** );
int PASCAL WriteGreyRgb (HWND hwnd,char *szFileName,MYBITMAP
*Bmap);
int PASCAL add_clean_points(HWND, UINT , UINT , LONG );
BOOL FAR PASCAL _export FramesDig(HWND ,UINT ,UINT ,LONG );
UINT FAR PASCAL _export TimerProc(HWND, UINT,UINT,LONG);
int PASCAL SonySearch(HWND,int , int );
int PASCAL SonyRecord(HWND, int);
int PASCAL load_field_from_card(HWND,int);
int PASCAL load_picture_from_card(HWND);
int PASCAL load_picture_from_file(HWND ,char *);
int PASCAL draw_field_to_screen(HWND ,MYBITMAP *.int );
int PASCAL WriteSign (HWND ,char *,SIGN *);
int PASCAL ReadSign(HWND ,char *,SIGN *);
int PASCAL pick_corners(HWND , UINT , UINT , LONG );
int PASCAL get_fitness_of_vertexes(HWND,RECT ,MYBITMAP *,RPOINT
*,DWORD *);
int PASCAL replace_sign(HWND, RPOINT *);

```

250



```

int PASCAL create_subs_sign(HWND hwnd) ;
int PASCAL fill_mask_rect(MYBITMAP *) ;
int PASCAL ValidateOccRects(MYBITMAP *) ;
int PASCAL copy_into_valid_rects() ;

char szAppName [] = "Trans" ;
static BYTE huge * lpRgb ;
static BYTE huge * lpGreyRgb ;
static BYTE huge * lpPtr ;
static BYTE huge * lpRepaint ;
static BYTE huge * lpDisplay ;
static BYTE huge * lpBits ;
PALETTEENTRY FAR *ppalentr ;
PALETTEENTRY FAR ppalsys[256] ;
BYTE FAR ColorTable[32][32][32] ;
HPALETTE glhpal, hpalprev ;
LOGPALETTE lgcpal ;
RECT GIBound ;
WORD glModelColor = COLOR_MODEL ;
char stBufferZoom[50] ;
char stBufferAngle[50] ;
char stBufferWidth[50] ;
int OrgX[8], OrgY[8] ;
int SignX[4], SignY[4] ;
char stBufferHeight[50] ;
char stBufferFrTop[50] ;
char stBufferFrBot[50] ;
RPOINT SrcPoly[4], DstPoly[4], KeepPoly[4] ;
RPOINT BstDst[4] ;
int InterpMode = IN_BILINEAR ;
MYBITMAP FAR *SrcBmap=NULL ;
MYBITMAP FAR *SrcLumap=NULL ;
MYBITMAP FAR *SubstBmap=NULL ;
MYBITMAP FAR *Subs=NULL ;
MYBITMAP FAR *Dispmap=NULL ;
MYBITMAP FAR *Normap=NULL ;
MYBITMAP FAR *MskBmap=NULL ;
SIGN OriginSign ;
RECT ClientRect ;
int lsRgb ;
int FileType ;
int SubstCols, SubstRows ;
int OrigCols, OrigRows ;
int CurrP, UnderPicking, UnderChanging, UnderTracking,
UnderOrig, UnderCorners ;
int InClean ;
BYTE huge* lpSubst ;
BYTE rpixarr[768] ;
TRACK_POINTS TrackBase, TrackCurr ;

```

```

Perspective_Transform BasicTransf;
Perspective_Transform ShiftTransf;
int CleanX[2];
int CleanY[2];
int CleanCount;
static FARPROC lpfTimeProc;
static LONG RecFrame = 20908;
static int SearchFrame = 15662;
int TimeElapsed;
static int PortNumber;
int FromFr;
static int ToFr = UNTIL_PICT;
int PictCount = 0;
int IsOddLine = 0;
static DWORD Ticks;
int WasInit = 0;
int SplitMode = 0;
OFSTRUCT of;
HFILE hFile;
int NumberOfField = 0;
double Vert1[4], Vert2[4], Vert3[4], Vert4[4];
double VertWeight[] = {2.0,2.0,3.0,3.0};
static int InSerieMode = 0;
static int ReplaceFlag = 1;
int DummyFrames = 0;
#define DYMMIES 0
RECT ValidOccRects[5];
int NumOfValidOccRects;
RECT OcclusionRects[5];
int NumOfOccRects;

int PASCAL WinMain (HANDLE hInstance, HANDLE hPrevInstance,
                    LPSTR lpszCmdLine, int nCmdShow)
{
    HWND hwnd;
    MSG msg;
    WNDCLASS wndclass;

    if (!hPrevInstance)
    {
        wndclass.style = CS_HREDRAW | CS_VREDRAW;
        wndclass.lpfWndProc = WndProc;
        wndclass.cbClsExtra = 0;
        wndclass.cbWndExtra = 0;
        wndclass.hInstance = hInstance;
        wndclass.hIcon = NULL;
        wndclass.hCursor = LoadCursor (NULL, IDC_ARROW);
        wndclass.hbrBackground = GetStockObject (WHITE_BRUSH);
        wndclass.lpszMenuName = szAppName;
    }

```

252

```

    wndclass.lpszClassName = szAppName ;

    RegisterClass (&wndclass) ;
}

hwnd = CreateWindow (szAppName, "Transformations",
    WS_OVERLAPPEDWINDOW,
    CW_USEDEFAULT, CW_USEDEFAULT,
    CW_USEDEFAULT, CW_USEDEFAULT,
    NULL, NULL, hInstance, NULL) ;

ShowWindow (hwnd, nCmdShow) ;
UpdateWindow (hwnd) ;

while (GetMessage (&msg, NULL, 0, 0))
{
    TranslateMessage (&msg) ;
    DispatchMessage (&msg) ;
}
return msg.wParam ;
}

long FAR PASCAL _export WndProc (HWND hwnd, UINT message, UINT
wParam,
                                LONG lParam)
{
    static char    szFileName    [_MAX_PATH],
                  szTitleName [_MAX_FNAME + _MAX_EXT] ;
    static char *  szFilter[] = { "RGB Files (*.RGB)",    "*.rgb",
                                "" } ;

    static OPENFILENAME ofn ;
    static FARPROC  lpfnFramesProc ;
    static HANDLE hinst ;

    HDC          hdc ;
    PAINTSTRUCT  ps ;
    char          DebugString[100] ;
    WORD          cxClient, cyClient ;
    HPALETTE      hpal ;
    DWORD         ij ;
    int           k ;
    char          RgbFileName[50] ;
    RECT          ToValidate, recp ;
    DWORD         Size ;
    int           DstRows, DstCols ;
    int           ToCopy ;
    int           gaoi, raoi, baoi ;
    PHDR          header ;

```

```

BYTE      huge * PtrRgb ;
char      DataBuffer[10], InputBuffer[10] ;
DCB       dcb ;
static int kernel[] = {
    1,1,1,
    1,1,1,
    1,1,1 };

switch (message) {
case WM_CREATE:
    ofn.lStructSize = sizeof (OPENFILENAME) ;
    ofn.hwndOwner = hwnd ;
    ofn.lpstrFilter = szFilter [0] ;
    ofn.lpstrFile = szFileName ;
    ofn.nMaxFile = _MAX_PATH ;
    ofn.lpstrFileTitle = szTitleName ;
    ofn.nMaxFileTitle = _MAX_FNAME + _MAX_EXT ;
    ofn.lpstrDefExt = "bmp" ;
    hinst = ((LPCREATESTRUCT)lParam)->hInstance ;
    lpfnFramesProc =
MakeProcInstance((FARPROC)FramesDlg,hinst);
    lpfnTimeProc = MakeProcInstance((FARPROC)TimerProc,hinst);
    lgcpal.palNumEntries = 256 ;
    lgcpal.palVersion = 0x300 ;
    lgcpal.palPalEntry[0].peRed = 0;
    lgcpal.palPalEntry[0].peGreen = 0;
    lgcpal.palPalEntry[0].peBlue = 0;
    lgcpal.palPalEntry[0].peFlags = PC_NOCOLLAPSE;
    hFile = OpenFile("map.dif",&of,OF_CREATE | OF_WRITE);
    OriginSign.Bmap = NULL ;
    OriginSign.BmapGrey = NULL ;
    return 0 ;
case WM_SIZE:
    cxClient = LOWORD (lParam) ;
    cyClient = HIWORD (lParam) ;
    return 0 ;
case WM_COMMAND:

    switch (wParam) {
    case IDM_OPEN:
        ofn.lpstrInitialDir = "c:\\work\\trans";
        if (GetOpenFileName (&ofn)) {
            FileType = get_file_type_by_name(szFileName) ;
            if ( FileType == FILE_IS_RGB ) {
                load_picture_from_file(hwnd,szFileName);
                InvalidateRect (hwnd, NULL, TRUE) ;
            }
        }
        break ;

```

254

SUBSTITUTE SHEET (RULE 26)

```

case IDM_WRITE:
    ofn.lpstrInitialDir = "c:\\work\\trans";
    if (GetOpenFileName (&ofn)) {
        FileType = get_file_type_by_name(szFileName);
        if (FileType == FILE_IS_RGB) {
            WriteRgb (hwnd,szFileName,SrcBmap);
        } else {
            MessageBox (hwnd, "File must be .rgb",
                szAppName, MB_ICONEXCLAMATION |
MB_OK);
        }
    }
    break;
case IDM_PICK_SUBST:
    if ( SrcBmap == NULL ) {
        MessageBox (hwnd, "No Source Bitmap",
            szAppName, MB_ICONEXCLAMATION |
MB_OK);
    }
    break;
    UnderPicking = 1;
    CurrP = 0;
    SetCapture(hwnd);
    SetCursor(LoadCursor(NULL, IDC_CROSS));
    break;
case IDM_PICK_ORIG:
    if ( SrcBmap == NULL ) {
        MessageBox (hwnd, "No Source Bitmap",
            szAppName, MB_ICONEXCLAMATION |
MB_OK);
    }
    break;
    UnderOrig = 1;
    CurrP = 0;
    SetCapture(hwnd);
    SetCursor(LoadCursor(NULL, IDC_CROSS));
    break;
case IDM_PICK_CORNERS:
    if ( SrcBmap == NULL ) {
        MessageBox (hwnd, "No Source Bitmap",
            szAppName, MB_ICONEXCLAMATION |
MB_OK);
    }
    break;
    UnderCorners = 1;
    CurrP = 0;
    SetCapture(hwnd);
    SetCursor(LoadCursor(NULL, IDC_CROSS));
    break;

```

```

case IDM_LOAD_ORIG:
    ReadSign( hwnd,"orig.sgn",&OriginSign) ;
    break ;
case IDM_CHANGE:
    UnderChanging = 1 ;
    if ( lpSubst == NULL ) {
        MessageBox (hwnd, "No Model Bitmap",
            szAppName, MB_ICONEXCLAMATION |
MB_OK) ;
        break ;
    }
    SetCapture(hwnd) ;
    SetCursor(LoadCursor(NULL, IDC_CROSS)) ;
    break ;
case IDM_BILINEAR:
    InterpMode = IN_BILINEAR ;
    break ;
case IDM_TRILINEAR:
    InterpMode = IN_TRILINEAR ;
    break ;
case IDM_SPLIT:
    SplitMode = 1 - SplitMode ;
    break ;
case IDM_INFO:
    display_information(hwnd) ;
    break ;
case IDM_START_TRACK:
    UnderTracking = 1 ;
    OriginSign.TrPoints.NumOfPoints = 0 ;
    TrackBase.NumOfPoints = 0 ;
    TrackCurr.NumOfPoints = 0 ;
    SetCapture(hwnd) ;
    SetCursor(LoadCursor(NULL, IDC_CROSS)) ;
    break ;
case IDM_AUTO_TRACK:
    //Ticks = GetTickCount() ;
    change_sign_by_tracking(hwnd,1) ;
    //sprintf(DebugString,"Track+Change %ld",
    //    GetTickCount()-Ticks) ;
    //MessageBox (hwnd, DebugString,
    //    szAppName, MB_ICONEXCLAMATION |
MB_OK) ;
    break ;
case DISP_INIT:
    mfg_loadcnf("");
    mfg_init() ;
    WasInit = 1 ;
    break ;
case DISP_SNAP:

```

256

SUBSTITUTE SHEET. (RULE 26)

```

        if ( WasInit == 0 ) {
            MessageBox (hwnd, "Need to Initialize MFG",
                szAppName, MB_ICONEXCLAMATION |
MB_OK);
            break ;
        }
        mfg_setvframe(RGB) ;
        mfg_dacmode(TRUE_24);
        mfg_snap(CAMERA,PAGE1) ;
        break ;
    case DISP_GRAB:
        if ( WasInit == 0 ) {
            MessageBox (hwnd, "Need to Initialize MFG",
                szAppName, MB_ICONEXCLAMATION |
MB_OK);
            break ;
        }
        mfg_setvframe(RGB) ;
        mfg_dacmode(TRUE_24);
        mfg_grab(CAMERA,PAGE1) ;
        break ;
    case DISP_DRAW:
        if ( WasInit == 0 ) {
            MessageBox (hwnd, "Need to Initialize MFG",
                szAppName, MB_ICONEXCLAMATION |
MB_OK);
            break ;
        }
        draw_to_screen(SrcBmap,0,SrcBmap->rows) ;
        //draw_to_screen(SrcBmap,120,200) ;
        break ;
    case DISP_CLEAN:
        if ( WasInit == 0 ) {
            MessageBox (hwnd, "Need to Initialize MFG",
                szAppName, MB_ICONEXCLAMATION |
MB_OK);
            break ;
        }
        SetCapture(hwnd) ;
        SetCursor(LoadCursor(NULL,IDC_CROSS)) ;
        InClean = 1 ;
        break ;
    case DISP_WIPE:
        if ( WasInit == 0 ) {
            MessageBox (hwnd, "Need to Initialize MFG",
                szAppName, MB_ICONEXCLAMATION |
MB_OK);
            break ;
        }
        break ;
    }

```

```

        mfg_wipe(0) ;
        InSerieMode = 1 ;
        //create_subs_sign(hwnd) ;
        SetTimer(hwnd,1,500,lpfnTimeProc) ; // To Be

REMOVED

        break ;
case DISP_LOAD:
    if ( WasInit == 0 ) {
        MessageBox (hwnd, "Need to Initialize MFG",
            szAppName, MB_ICONEXCLAMATION |

MB_OK) ;

        break ;
    }
    //load_picture_from_card(hwnd) ;
    sprintf(RgbFileName,"pict%d.rgb",PictCount++) ;
    WriteRgb (hwnd,RgbFileName,SrcBmap);
    InvalidateRect (hwnd, NULL, TRUE) ;
    break ;
case DISP_LOAD_FIELD:
    if ( WasInit == 0 ) {
        MessageBox (hwnd, "Need to Initialize MFG",
            szAppName, MB_ICONEXCLAMATION |

MB_OK) ;

        break ;
    }
    load_field_from_card(hwnd,0) ;
    InvalidateRect (hwnd, NULL, TRUE) ;
    break ;
case SONY_INIT:
    PortNumber = OpenComm("COM2",1024,128) ;
    sprintf(DebugString,"PortNumber %d",PortNumber) ;
    MessageBox (hwnd, DebugString,
        szAppName, MB_ICONEXCLAMATION | MB_OK) ;
    k= BuildCommDCB("COM2:1200,n,8,1",&dcb) ;
    k = SetCommState(&dcb) ;
    break ;
case SONY_FRAME:
    if ( DialogBox(hinst,"frames",hwnd,lpfnFramesProc) ) {
        sscanf(stBufferWidth,"%d",&FromFr) ;
        sscanf(stBufferHeight,"%d",&ToFr) ;
    }
    //SonySearch(hwnd,FromFr,PortNumber) ;
    //TimeElapsed = 0 ;
    //SetTimer(hwnd,1,500,lpfnTimeProc) ;
    break ;
case SONY_RECORD:
    ReplaceFlag = 1 - ReplaceFlag ;
    //SonyRecord(hwnd,PortNumber) ;
    break ;

```



```

        case SONY_END:
            k = CloseComm(PortNumber);
            break;
    }
    break;
case WM_LBUTTONDOWN:
    if ( UnderPicking == 1 ) {
        pick_sign(hwnd,message,wParam,lParam);
    }
    if ( UnderOrig == 1 ) {
        pick_original(hwnd,message,wParam,lParam);
    }
    if ( UnderCorners == 1 ) {
        pick_corners(hwnd,message,wParam,lParam);
    }
    if ( UnderChanging == 1 ) {
        change_sign(hwnd,message,wParam,lParam);
    }
    if ( UnderTracking == 1 ) {
        add_tracking_point(hwnd,message,wParam,lParam);
    }
    if ( InClean == 1 ) {
        add_clean_points(hwnd,message,wParam,lParam);
    }
    return 0;
case WM_RBUTTONDOWN:
    if ( UnderTracking == 1 ) {
        SetCursor(LoadCursor(NULL, IDC_ARROW));
        ReleaseCapture();
        UnderTracking = 0;
        create_lum_bmap(OriginSign.Bmap, &(OriginSign.BmapGrey));
        WriteSign(hwnd,"orig.sgn",&OriginSign);
    }
    return 0;
case WM_MOUSEMOVE:
    if ( UnderPicking == 1 || UnderChanging == 1 || UnderOrig == 1 )
    {
        sprintf(DebugString,"%d,%d",LOWORD(lParam),HIWORD(lParam));
        recp.left = recp.top = 400;
        recp.bottom = 420;
        recp.right = 480;
        InvalidateRect(hwnd,&recp,TRUE);
    }
    return 0;
case WM_PAINT:
    if ( InSerieMode == 1 ) {
        hdc = BeginPaint (hwnd, &ps);
        EndPaint (hwnd, &ps);
    }

```

254

SUBSTITUTE SHEET (RULE 26)

```

        break ;
    }
    hdc = BeginPaint (hwnd, &ps) ;
    ToValidate = ps.rcPaint ;
    if ( UnderPicking == 1 || UnderChanging == 1 || UnderOrig == 1 ) {
        TextOut(hdc, ToValidate.left, ToValidate.top, DebugString, lstrlen(DebugString))
        ;
    } else {
        if ( SrcBmap != NULL ) {
            glhpal = CreatePalette(&lgcpal) ;
            if ( glhpal == NULL )
                MessageBox (hwnd, "PALLET NOT CREATED",
                szAppName, MB_ICONEXCLAMATION | MB_OK) ;
            SetPaletteEntries(glhpal, 0, 256, ppalentr) ;
            hpalprev = SelectPalette(hdc, glhpal, FALSE) ;
            RealizePalette(hdc) ;
            if ( bitmap_for_display(hwnd, hdc, SrcBmap, ToValidate) == 0 ) {
                DeleteObject(glhpal) ;
                EndPaint (hwnd, &ps) ;
                return 0 ;
            }
            SetStretchBltMode(hdc, COLORONCOLOR) ;
            SetDIBitsToDevice(hdc, 0, 0, Dispmap->cols,
                Dispmap->rows, 0, 0, 0, Dispmap->rows,
                (LPSTR)(Dispmap->gpics),
                (LPBITMAPINFO)lpDisplay, DIB_RGB_COLORS) ;
            DeleteObject(glhpal) ;
        }
    } // End Of Else
    EndPaint (hwnd, &ps) ;
    return 0 ;

case WM_DESTROY:
    _lclose(hFile) ;
    if ( lpDisplay != NULL ) {
        GlobalFreePtr(lpDisplay) ;
    }
    bm_free(SrcBmap) ;
    bm_free(SrcLumap) ;
    bm_free(Dispmap) ;
    bm_free(Normap) ;
    bm_free(SubstBmap) ;
    bm_free(Subs) ;
    bm_free(MskBmap) ;
    KillTimer(hwnd, 1) ;
    PostQuitMessage (0) ;
    return 0 ;
}

```

260

```

    return DefWindowProc (hwnd, message, wParam, lParam) ;
}

int PASCAL teach_grey_pallette(HWND hwnd, int Entries)
{
    int i;
    UINT rc;
    char Debug[100];

    if ( ppalentr != NULL ) {
        GlobalFreePtr(ppalentr);
        ppalentr = NULL;
    }
    ppalentr = (PALETTEENTRY FAR *) GlobalAllocPtr (GMEM_MOVEABLE
        ,Entries*sizeof(RGBQUAD));

    for ( i = 0; i < Entries; i++) {
        ppalentr[i].peRed = i;
        ppalentr[i].peGreen = i;
        ppalentr[i].peBlue = i;
        ppalentr[i].peFlags = PC_NOCOLLAPSE;
    }

    return 1;
}

int PASCAL get_in_series_flag()
{
    return InSerieMode;
}

int PASCAL display_information(HWND hwnd)
{
    char Debug[100];

    if ( SrcBmap == NULL ) {
        MessageBox (hwnd, "No Source Bitmap", szAppName,
            MB_ICONEXCLAMATION | MB_OK);

        return 1;
    }

    sprintf(Debug, "Width = %d Height = %d ", SrcBmap->cols, SrcBmap->rows);
    MessageBox (hwnd, Debug, szAppName, MB_ICONEXCLAMATION |
        MB_OK);
    return 1;
}

```

```

BOOL FAR PASCAL _export FramesDlg(HWND hdlg,UINT message,UINT
wParam,

```

```

LONG lParam)

```

```

{
    switch ( message ) {
        case WM_INITDIALOG:
            return TRUE ;
        case WM_COMMAND:
            switch ( wParam ) {
                case SONY_FROM_FR:

                    GetDlgItemText(hdlg,PER_WIDTH,(LPSTR)stBufferWidth,50) ;
                    return TRUE ;
                    break ;
                case SONY_TO_FR:

                    GetDlgItemText(hdlg,PER_HEIGHT,(LPSTR)stBufferHeight,50) ;
                    return TRUE ;
                    break ;
                case SONY_OK:
                    EndDialog(hdlg,TRUE) ;
                    return TRUE ;
                    break ;
                case SONY_CANCEL:
                    EndDialog(hdlg,FALSE) ;
                    return TRUE ;
                    break ;
            }
    }
    return FALSE ;
}

```

```

BYTE huge * ReadRGB(HWND hwnd,char * FileName,int *Width,int *Height)
{
    HFILE hFile ;
    PHDR phdr;
    DWORD Size ;
    HDC hdc ;
    DWORD offset ;
    DWORD i,j ;
    BYTE Red,Green,Blue ;
    HPALETTE hpal ;
    int ColorFactor ;
    BYTE huge * PtrRgb ;
    BYTE huge * lpRead ;
    int SizeToRead ;

```

262

```

int Incr ;

    if (-1 == (hFile = _lopen (FileName, OF_READ |
    OF_SHARE_DENY_WRITE)))
        return NULL ;
    _lread (hFile, (LPSTR) &phdr, sizeof (PHDR)) ;
    if ( glModelColor == GREY_MODEL )      ColorFactor = 1 ;
    if ( glModelColor == COLOR_MODEL )    ColorFactor = 3 ;
    Size = (DWORD)phdr.cols*(DWORD)phdr.rows*(DWORD)ColorFactor ;
    *Width = phdr.cols ;
    *Height = phdr.rows ;
    lpRgb = (BYTE huge *) GlobalAllocPtr (GMEM_MOVEABLE, Size) ;
    PtrRgb = lpRgb ;
    offset = 0 ;
    SizeToRead = (DWORD)(phdr.cols)*(DWORD)3 ;
    lpRead = (BYTE huge *) GlobalAllocPtr (GMEM_MOVEABLE,
    SizeToRead) ;
    while ( offset < Size ) {
        if (glModelColor == GREY_MODEL) {
            _lread(hFile,(LPSTR)lpRead,SizeToRead) ;
            for ( i = 0 ; i < SizeToRead ; i+= 3 ) {
                *(PtrRgb+offset) = (*(lpRead+i)+*(lpRead+i+1)+
                *(lpRead+i+2))/3 ;
            }
            offset++ ;
        } else { // COLOR_MODEL
            _lread(hFile,(LPSTR)(PtrRgb+offset),SizeToRead) ;
            offset += SizeToRead ;
        }
    }
    GlobalFreePtr (lpRead) ;
    if (glModelColor == GREY_MODEL) {
        teach_grey_pallete(hwnd, 256) ;
    } else {
        teach_rgb_pallete(hwnd) ;
    }
    _lclose(hFile) ;
    return PtrRgb ;
}

```

```

int PASCAL teach_rgb_pallete(HWND hwnd)
{
    int    pl,i,j,k ;
    int    red, blue, green ;
    UINT   rc ;
    char    Debug[100] ;
    int     Scale[6] = {0,64,128,150,192,255} ;
    int     Fine[4] = {32,96,171,224} ;

```

```

if ( ppalentr != NULL ) {
    GlobalFreePtr(ppalentr);
    ppalentr = NULL;
}
ppalentr = (PALETTEENTRY FAR *) GlobalAllocPtr (GMEM_MOVEABLE
                                                ,256*sizeof(RGBQUAD));
for ( i = 10 ; i < 245 ; i++ ) {
    ppalentr[i].peRed = i;
    ppalentr[i].peGreen = i;
    ppalentr[i].peBlue = i;

    ppalentr[i].peFlags = PC_RESERVED;
}

return 1;
}

```

```

int PASCAL get_file_type_by_name(char * FileName)
{
    int i,len;

    len = strlen(FileName)-1;
    for ( i = len ; i > 0 ; i-- ) {
        FileName[i] = toupper(FileName[i]);
        if ( FileName[i] == '.' ) {
            i++;
            break;
        }
    }
    if ( strcmp(FileName+i,"RGB") == 0 ) return FILE_IS_RGB;
    return FILE_IS_UNKNOWN;
}

```

```

int PASCAL create_poly_src_dst()
{
    SrcPoly[0].x = SrcPoly[0].x = 0.0;
    SrcPoly[1].x = SubstBmap->cols; // SubstCols;
    SrcPoly[1].y = 0.0;
    SrcPoly[2].x = SubstBmap->cols; //SubstCols;
    SrcPoly[2].y = SubstBmap->rows; //SubstRows;
    SrcPoly[3].x = 0.0;
    SrcPoly[3].y = SubstBmap->rows; //SubstRows;

    I_cp_int_arr_to_RPOINT(DstPoly, OrgX, OrgY,4);
    return 1;
}

```

264

```

}
```

```

int PASCAL keep_subst(HWND hwnd, MYBITMAP *Bmap)
```

```

{
    DWORD i,j;
    HFILE hFile;
    PHDR phdr;
    DWORD Size;
    DWORD offset;
    DWORD Cols;
    BYTE huge * Ptr;
    DWORD Width, Adjust;

    SubstCols = phdr.cols = abs(OrgX[1] - OrgX[0]);
    SubstRows = phdr.rows = abs(OrgY[2] - OrgY[1]);
    phdr.bp = 3;
    Size = (DWORD)(phdr.cols)*(DWORD)(phdr.rows)*(DWORD)3;
    lpSubst = (BYTE huge*)GlobalAllocPtr(GMEM_MOVEABLE, Size);
    offset = 0;
    Cols = (DWORD)(phdr.cols)*(DWORD)3;
    Ptr = Bmap->gpics;

    Width = (DWORD)(SrcBmap->cols)*(DWORD)3;
    Adjust = (DWORD)(OrgX[0])*(DWORD)3;

    for ( i = 0 ; i < phdr.rows ; i++ ){
        for ( j = 0 ; j < Cols ; j++ ) {
            *(lpSubst + offset+j) = *(Ptr+
                ((DWORD)Width*(DWORD)(OrgY[0]+i)) + Adjust + j);
        }
        offset += Cols;
    }
    SubstBmap = bm_alloc(SubstCols, SubstRows, glModelColor);
    SubstBmap->gpics = lpSubst;
    return 1;
}
```

```

int PASCAL keep_orig(HWND hwnd, MYBITMAP *Bmap)
```

```

{
    DWORD i,j;
    HFILE hFile;
    PHDR phdr;
    DWORD Size;
    DWORD offset;
    BYTE huge* lpModel;
    DWORD Cols;
    BYTE huge * Ptr;
    DWORD Width, Adjust;
```

```

OrigCols = phdr.cols = abs(OrgX[1] - OrgX[0]) ;
OrigRows = phdr.rows = abs(OrgY[2] - OrgY[1]) ;
phdr.bp = 24 ;
Size = (DWORD)(phdr.cols)*(DWORD)(phdr.rows)*(DWORD)3 ;
lpModel = (BYTE huge*)GlobalAllocPtr(GMEM_MOVEABLE,Size) ;
offset = 0 ;
Cols = (DWORD)(phdr.cols)*(DWORD)3 ;
Ptr = Bmap->gpic ;

Width = (DWORD)(SrcBmap->cols)*(DWORD)3 ;
Adjust = (DWORD)(OrgX[0])*(DWORD)3 ;

for ( i = 0 ; i < phdr.rows ; i++ ) {
    Ptr = Bmap->gpic + ((DWORD)Width*(DWORD)(OrgY[0]+i)) + Adjust ;
    for ( j = 0 ; j < Cols ; j++ ) {
        *(lpModel + offset+j) = *(Ptr++) ;
    }
    offset += Cols ;
}
bm_free(OriginSign.Bmap) ;
bm_free(OriginSign.BmapGrey) ;
OriginSign.Bmap = bm_alloc(OrigCols,OrigRows,glModelColor);
OriginSign.Bmap->gpic = lpModel ;
WriteRgb (hwnd,"orig.rgb",OriginSign.Bmap);
return 1 ;
}

```

```

int PASCAL bitmap_for_display(HWND hwnd, HDC hdc,MYBITMAP FAR
*Bmap,RECT ToValidate)

```

```

{
    LONG Size,i,j ;
    BYTE huge * TmpB ;
    BYTE huge * Tmp ;
    BYTE huge * Ptr ;
    char Debug[100] ;

```

```

    GetSystemPaletteEntries(hdc,(UINT)0,(UINT)256,ppalsys) ;
    Size = 40 +(DWORD)(sizeof(RGBQUAD))*(DWORD)256 ;

```

```

    if ( lpDisplay != NULL ) {
        GlobalFreePtr(lpDisplay) ;
    }

```

```

    lpDisplay = (BYTE huge*)GlobalAllocPtr(GMEM_MOVEABLE,Size) ;
    lpBits = lpDisplay + 40+(DWORD)(sizeof(RGBQUAD))*(DWORD)256 ;
    Ptr = lpDisplay + 40 ;

```

266



```

((BITMAPINFOHEADER huge*)lpDisplay)->biSize = 40 ;
((BITMAPINFOHEADER huge*)lpDisplay)->biWidth = Bmap->cols ;
((BITMAPINFOHEADER huge*)lpDisplay)->biHeight = Bmap->rows ;
((BITMAPINFOHEADER huge*)lpDisplay)->biPlanes = 1 ;
((BITMAPINFOHEADER huge*)lpDisplay)->biBitCount = 8 ;
((BITMAPINFOHEADER huge*)lpDisplay)->biCompression = 0 ;
((BITMAPINFOHEADER huge*)lpDisplay)->biSizeImage = 0 ;
((BITMAPINFOHEADER huge*)lpDisplay)->biXPelsPerMeter = 0 ;
((BITMAPINFOHEADER huge*)lpDisplay)->biYPelsPerMeter = 0 ;
((BITMAPINFOHEADER huge*)lpDisplay)->biClrUsed = 0 ;
((BITMAPINFOHEADER huge*)lpDisplay)->biClrImportant = 0 ;

for ( i = 0 ; i < 256 ; i++ ) {
    (BYTE)*(Ptr + (DWORD)(i*sizeof(RGBQUAD))) = ppalsys[i].peBlue;
    (BYTE)*(Ptr + (DWORD)(i*sizeof(RGBQUAD)+1))= ppalsys[i].peGreen;
    (BYTE)*(Ptr + (DWORD)(i*sizeof(RGBQUAD)+2))= ppalsys[i].peRed;
    (BYTE)*(Ptr + (DWORD)(i*sizeof(RGBQUAD)+3)) = 0 ;
}
return (create_disp_bmap(Bmap, &Dispmap, ToValidate)) ;
}

```

```

int PASCAL draw_to_screen(MYBITMAP *Bmap,int FromLine,int ToLine)

```

```

{
    int i,j;
    BYTE huge *Ptr ;
    BYTE huge *PtrR ;
    BYTE huge *PtrG ;
    BYTE huge *PtrB ;
    int Cols ;

    Cols = Bmap->cols ;
    Ptr = Bmap->gpics ;
    Ptr += (DWORD)FromLine*(DWORD)Cols*(DWORD)3 ;
    PtrR = Ptr ;
    PtrG = Ptr+1 ;
    PtrB = Ptr+2 ;
    mfg_setgframe(R);
    for ( i = FromLine ; i < ToLine ; i++ ) {
        for ( j = 0 ; j < Cols ; j++ ) {
            rpixarr[j] = *PtrR ;
            PtrR += 3 ;
        }
        mfg_bwhline(PAGE0,0,i,Cols,rpixarr) ;
    }
    mfg_setgframe(G);
    for ( i = FromLine ; i < ToLine ; i++ ) {
        for ( j = 0 ; j < Cols ; j++ ) {
            rpixarr[j] = *PtrG ;
            PtrG += 3 ;

```

267

SUBSTITUTE SHEET (RULE 26)

```

    }
    mfg_bwhline(PAGE0,0,i,Cols,rpixarr);
}
mfg_setgframe(B);
for ( i = FromLine; i < ToLine; i++ ) {
    for ( j = 0; j < Cols; j++ ) {
        rpixarr[j] = *PtrB;
        PtrB += 3;
    }
    mfg_bwhline(PAGE0,0,i,Cols,rpixarr);
}
return 1;
}

int PASCAL create_disp_bmap(MYBITMAP *Bmap, MYBITMAP
**LumBmap,RECT ToValidate)
{
    BYTE huge * Tmp;
    BYTE huge * TmpB;
    int ToPad;
    int Cols, Rows;
    DWORD Size;
    int i,j;
    long k;
    int RowLimit;

    if ( Normap == NULL ) return 0;
    Cols = Bmap->cols;
    Rows = Bmap->rows;

    ToPad = Cols%4;
    if ( ToPad > 0 ) ToPad = 4 - ToPad;

    Tmp = Bmap->gpics;

    if ( ToValidate.top == 0 ) {
        if ( *LumBmap != NULL ) bm_free(*LumBmap);

        *LumBmap = bm_alloc(Cols+ToPad,Rows,GREY_MODEL);
        Size = (DWORD)(Cols+ToPad)*(DWORD)Rows;
        (*LumBmap)->gpics = (BYTE
huge*)GlobalAllocPtr(GMEM_MOVEABLE,Size);
    }

    RowLimit = min(Rows,ToValidate.bottom);
    for ( i = RowLimit; i > ToValidate.top; i-- ) {
        Tmp = Normap->gpics + (DWORD)(i-1)*(DWORD)Cols;

```

263

SUBSTITUTE SHEET (RULE 26)

```

    TmpB = (*LumBmap)->gpic+(Rows-i)*(DWORD)(Cols+ToPad) ;
    for ( j = 0 ; j < Cols ; j++ ) {
        *(TmpB++) = *(Tmp++) ;
    }
    for ( j = 0 ; j < ToPad ; j++ ) {
        *(TmpB++) = 0 ;
    }
}
return 1 ;
}

```

```

int PASCAL pick_sign(HWND hwnd, UINT message, UINT wParam, LONG
IParam)
{

```

```

    OrgX[CurrP] = LOWORD(IParam);
    OrgY[CurrP] = HIWORD(IParam);
    CurrP++;
    if ( CurrP == 2 ) {
        OrgX[2] = OrgX[1];
        OrgY[2] = OrgY[1];
        OrgX[1] = OrgX[0];
        OrgY[1] = OrgY[0];
        OrgX[3] = OrgX[0];
        OrgY[3] = OrgY[2];
        UnderPicking = 0;
        CurrP = 0;
        SetCursor(LoadCursor(NULL, IDC_ARROW));
        ReleaseCapture();
        keep_subst(hwnd, SrcBmap);
        if ( InterpMode == IN_TRILINEAR )
            build_pyramid(hwnd, SubstBmap);
    }

    return 1;
}

```

```

int PASCAL add_clean_points(HWND hwnd, UINT message, UINT wParam,
LONG IParam)
{

```

```

    static int kernel[] = {
        1,1,1,
        1,1,1,
        1,1,1 };
    int gaoi,raoi,baoi ;
    int DifX,DifY ;

```

```

    CleanX[CleanCount] = LOWORD(IParam);
    CleanY[CleanCount] = HIWORD(IParam);
    CleanCount++;

```

264

```

    if ( CleanCount == 2 ) {
        InClean = 0 ;
        CleanCount = 0 ;
        SetCursor(LoadCursor(NULL, IDC_WAIT)) ;
        DifX = CleanX[1]-CleanX[0] ;
        DifY = CleanY[1]-CleanY[0] ;
        gaoi = mfg_gaoi_fbcreate(G, CleanX[0], CleanY[0], DifX, DifY) ;
        mfg_median(gaoi, gaoi, 3, 3, kernel) ;
        raoi = mfg_gaoi_fbcreate(R, CleanX[0], CleanY[0], DifX, DifY) ;
        mfg_median(raoi, raoi, 3, 3, kernel) ;
        baoi = mfg_gaoi_fbcreate(B, CleanX[0], CleanY[0], DifX, DifY) ;
        mfg_median(baoi, baoi, 3, 3, kernel) ;
        SetCursor(LoadCursor(NULL, IDC_ARROW)) ;
        ReleaseCapture() ;
        //SetTimer(hwnd, 1, 40, lpfnTimeProc) ;
    }

    return 1 ;
}

int PASCAL pick_corners(HWND hwnd, UINT message, UINT wParam,
LONG lParam)
{
    RPOINT CoefsH1, CoefsH2, CoefsV1, CoefsV2 ;
    POINT Point1, Point2 ;
    char String[100] ;

    OrgX[CurrP] = LOWORD(lParam);
    OrgY[CurrP] = HIWORD(lParam);
    CurrP++ ;
    if ( CurrP == 4 ) {
        UnderCorners = 0 ;
        CurrP = 0 ;
        SetCursor(LoadCursor(NULL, IDC_ARROW)) ;
        ReleaseCapture() ;
    }

    return 1 ;
}

int PASCAL pick_original(HWND hwnd, UINT message, UINT wParam,
LONG lParam)
{
    RPOINT CoefsH1, CoefsH2, CoefsV1, CoefsV2 ;
    POINT Point1, Point2 ;
    char String[100] ;

    OrgX[CurrP] = LOWORD(lParam);

```

```

    OrgY[CurrP] = HIWORD(IParam);
    CurrP++;
    if ( CurrP == 2 ) {
        OrgX[2] = OrgX[1];
        OrgY[2] = OrgY[1];
        OrgY[1] = OrgY[0];
        OrgX[3] = OrgX[0];
        OrgY[3] = OrgY[2];
        UnderOrig = 0;
        CurrP = 0;
        SetCursor(LoadCursor(NULL, IDC_ARROW));
        ReleaseCapture();
        keep_orig(hwnd, SrcBmap);
    }

    return 1;
}

int PASCAL change_sign(HWND hwnd, UINT message, UINT wParam,
LONG lParam)
{
    RECT          Bound;
    RPOINT Pt0,Pt1,Pt2,Pt3;
    POINT Point1,Point2;
    RPOINT SrcPnts[4];
    char String[100];
    DWORD FromTime,ToTime;
    EDGE Edge1,Edge2,Edge3,Edge4;
    char DebugString[50];
    MYBITMAP *AlphaMap;

    OrgX[CurrP] = LOWORD(IParam);
    OrgY[CurrP] = HIWORD(IParam);
    CurrP++;
    if ( CurrP == 4 ) {
        UnderChanging = 0;
        CurrP = 0;
        SetCursor(LoadCursor(NULL, IDC_ARROW));
        ReleaseCapture();

        create_poly_src_dst();
        l_find_bound_rect(DstPoly, &Bound);

        //create_mask_bmap(hwnd,OriginSign.BmapGrey,SrcPoly,DstPoly,Bound,&M
skBmap);
        //Ticks = GetTickCount();
        build_alpha_map(&AlphaMap, SrcBmap,DstPoly);

        //perspective_al(hwnd,SubstBmap,SrcBmap,AlphaMap,SrcPoly,
27!

```

```

        //      DstPoly, glModelColor, &BasicTransf) ;
        bm_free(AlphaMap) ;
        perspective(hwnd, SubstBmap, SrcBmap, SrcPoly,
            DstPoly, glModelColor, &BasicTransf) ;
        //sprintf(DebugString, "Change %ld", GetTickCount()-Ticks) ;
        //MessageBox (hwnd, DebugString,
        //      szAppName, MB_ICONEXCLAMATION | MB_OK) ;
        //perspective_mask(hwnd, SubstBmap, SrcBmap, SrcPoly,
        //      DstPoly, glModelColor, InterpMode, MskBmap) ;
        create_norm_bmap(hwnd, SrcBmap, &Normap, Bound) ;
        InvalidateRect (hwnd, &Bound, TRUE) ;
    }
    return 1 ;
}

int PASCAL add_tracking_point(HWND hwnd, UINT message, UINT wParam,
                                LONG lParam)
{
    int Place ;

    Place = OriginSign.TrPoints.NumOfPoints ;
    if ( Place >= NUM_OF_TRACK_POINTS ) {
        MessageBox (hwnd, "Cannot track more than 30 points",
            szAppName, MB_ICONEXCLAMATION | MB_OK) ;
        SetCursor(LoadCursor(NULL, IDC_ARROW)) ;
        ReleaseCapture() ;
        UnderTracking = 0 ;
        create_lum_bmap(OriginSign.Bmap, &(OriginSign.BmapGrey)) ;
        WriteSign(hwnd, "orig.sgn", &OriginSign) ;
        return 1 ;
    }
    OriginSign.TrPoints.TrackP[Place].x = LOWORD(lParam);
    OriginSign.TrPoints.TrackP[Place].y = HIWORD(lParam);
    TrackBase.TrackP[Place].x = LOWORD(lParam);
    TrackBase.TrackP[Place].y = HIWORD(lParam);
    TrackCurr.TrackP[Place].x = LOWORD(lParam);
    TrackCurr.TrackP[Place].y = HIWORD(lParam);
    OriginSign.TrPoints.NumOfPoints++ ;
    TrackBase.NumOfPoints++ ;
    TrackCurr.NumOfPoints++ ;

    return 1 ;
}

int PASCAL change_sign_by_tracking(HWND hwnd, int ToDisplay)
{
    Perspective_Transform NewTransf ;
    Perspective_Transform Tp ;

```

272

SUBSTITUTE SHEET (RULE 26)

```

Perspective_Transform TpCurr ;
RECT Bound ;
int i,k ;
double DiffUp,DiffDown ;
double u,v,w ;
char String[100] ;
int dx,dy ;
DWORD Size ;
int Cols,Rows ;
RPOINT SrcPnts[4] ;
RPOINT RPointIn,RPointOut ;
SHIFT Shift ;
SHIFT Shifts[NUM_OF_TRACK_POINTS];
double DstX,DstY ;
RECT Rectan ;
MYBITMAP *DestBmap ;
RPOINT CrnPoly[4] ;
TRACK_POINTS ModelPoints ;
TRACK_POINTS DBasePoints ;
RPOINT DiffArr[MAX_NO_OF_TR_WIN] ;
MYBITMAP *AlphaMap ;
TR_WIN Windows[MAX_NO_OF_TR_WIN] ;

//create_subs_sign(hwnd) ;

sprintf(String,"\\n\\n\\t\\tNew Frame\\n");
_lwrite(hFile,String,strlen(String));
_l_cp_int_arr_to_RPOINT(SrcPnts, SignX, SignY,4) ;

Quad2Quad(hwnd,SrcPnts,DstPoly,&TpCurr) ;

_l_find_bound_rect(DstPoly, &Rectan) ;
Cols = Rectan.right-Rectan.left+1 ;
Rows = Rectan.bottom-Rectan.top+1 ;
DestBmap = bm_alloc(Cols,Rows,GREY_MODEL) ;
Size = (DWORD)Cols*(DWORD)Rows ;
DestBmap->gpic = (BYTE huge*)GlobalAllocPtr(GMEM_MOVEABLE,Size) ;

_l_quad_in_new_origin(CrnPoly,DstPoly,Rectan.left,Rectan.top,4) ;
for ( i = 0 ; i < 4 ; i++ ) {
    sprintf(String,"%d: Dst %lf,%lf Crn %lf,%lf Src %lf,%lf\\n",
        i,DstPoly[i].x,DstPoly[i].y,CrnPoly[i].x,CrnPoly[i].y,
        SrcPnts[i].x,SrcPnts[i].y) ;
    _lwrite(hFile,String,strlen(String)) ;
}
Quad2Quad(hwnd,SrcPnts, CrnPoly,&Tp);

```

```

TrackCurr.NumOfPoints = 0 ;
ModelPoints.NumOfPoints = 0 ;
DBasePoints.NumOfPoints = 0 ;

for ( i = k = 0 ; i < OriginSign.TrPoints.NumOfPoints ; i++ ) {
    RPointIn.x = (double)OriginSign.TrPoints.TrackP[i].x ;
    RPointIn.y = (double)OriginSign.TrPoints.TrackP[i].y ;
    RPointOut = fPerspective(RPointIn, &Tp) ;
    if (RPointOut.x <= CORR_WINDOWX || RPointOut.x >= (Cols-
CORR_WINDOWX-1) ) continue ;
    if (RPointOut.y <= CORR_WINDOWY || RPointOut.y >= (Rows-
CORR_WINDOWY-1) ) continue ;
    ModelPoints.TrackP[k].x = (int)(RPointOut.x+0.5) ;
    ModelPoints.TrackP[k].y = (int)(RPointOut.y+0.5) ;
    if ( ModelPoints.TrackP[k].x >    RPointOut.x ) { // To minus DiffArr
        DiffArr[k].x = (double)ModelPoints.TrackP[k].x - RPointOut.x ;
    } else {
        DiffArr[k].x = RPointOut.x - (double)ModelPoints.TrackP[k].x ;
    }
    if ( ModelPoints.TrackP[k].y >    RPointOut.y ) {
        DiffArr[k].y = (double)ModelPoints.TrackP[k].y - RPointOut.y ;
    } else {
        DiffArr[k].y = RPointOut.y - (double)ModelPoints.TrackP[k].y ;
    }
    DBasePoints.TrackP[k].x = OriginSign.TrPoints.TrackP[i].x ;
    DBasePoints.TrackP[k].y = OriginSign.TrPoints.TrackP[i].y ;
    RPointOut = fPerspective(RPointIn, &TpCurr) ;
    TrackCurr.TrackP[k].x = (int)(RPointOut.x+0.5) ;
    TrackCurr.TrackP[k].y = (int)(RPointOut.y+0.5) ;
    if ( TrackCurr.TrackP[k].x >    RPointOut.x ) { // To minus DiffArr
        DiffArr[k].x += (double)TrackCurr.TrackP[k].x - RPointOut.x ;
    } else {
        DiffArr[k].x += RPointOut.x - (double)TrackCurr.TrackP[k].x ;
    }
    if ( TrackCurr.TrackP[k].y >    RPointOut.y ) {
        DiffArr[k].y += (double)TrackCurr.TrackP[k].y - RPointOut.y ;
    } else {
        DiffArr[k].y += RPointOut.y - (double)TrackCurr.TrackP[k].y ;
    }
    k++ ;
    TrackCurr.NumOfPoints++ ;
    DBasePoints.NumOfPoints++ ;
    ModelPoints.NumOfPoints++ ;
}
perspective(hwnd, OriginSign.BmapGrey, DestBmap, SrcPnts,
            CmnPoly, GREY_MODEL, &BasicTransf) ;

xysolve(hwnd, Normap, DestBmap, Shifts, Windows, 9, &ModelPoints,
        &TrackCurr, &NewTransf, hFile, &DBasePoints, DiffArr) ;

```



```

bm_free(DstBmap) ;
for ( i = 0 ; i < 4 ; i++ ) {
    sprintf(String,"Before %d --%lf,%lf\n",i,SrcPnts[i].x, SrcPnts[i].y) ;
    _lwrite(hFile,String,strlen(String)) ;
    //DstX = SrcPnts[i].x * NewTransf.Pf[0][0] +
    //      SrcPnts[i].y * NewTransf.Pf[1][0] + NewTransf.Pf[2][0] ;
    //DstY = SrcPnts[i].x * NewTransf.Pf[0][1] +
    //      SrcPnts[i].y * NewTransf.Pf[1][1] + NewTransf.Pf[2][1] ;
    //w = SrcPnts[i].x * NewTransf.Pf[0][2] +
    //      SrcPnts[i].y * NewTransf.Pf[1][2] + NewTransf.Pf[2][2] ;
    DstX = CmPoly[i].x * NewTransf.Pf[0][0] +
           CmPoly[i].y * NewTransf.Pf[1][0] + NewTransf.Pf[2][0] ;
    DstY = CmPoly[i].x * NewTransf.Pf[0][1] +
           CmPoly[i].y * NewTransf.Pf[1][1] + NewTransf.Pf[2][1] ;
    w = CmPoly[i].x * NewTransf.Pf[0][2] +
         CmPoly[i].y * NewTransf.Pf[1][2] + NewTransf.Pf[2][2] ;
    DstPoly[i].x = DstX/w ;
    DstPoly[i].y = DstY/w ;
    sprintf(String,"Vertex %d --%lf,%lf\n",i,DstPoly[i].x, DstPoly[i].y) ;
    _lwrite(hFile,String,strlen(String)) ;
}

l_find_bound_rect(DstPoly, &GIBound) ;
/* OCCLUSION

create_mask_bmap(hwnd,OriginSign.BmapGrey,SrcPnts,DstPoly,GIBound,&
MskBmap) ;
fill_mask_rect(MskBmap) ;
ValidateOccRects(MskBmap) ;
copy_into_valid_rects() ;
*/

//Ticks = GetTickCount() ;
build_alpha_map(&AlphaMap, SrcBmap,DstPoly) ;
//WriteGreyRgb (hwnd,"alp.rgb",AlphaMap) ;
perspective_al(hwnd,SubstBmap,SrcBmap,AlphaMap,SrcPoly,
               DstPoly, glModelColor, &BasicTransf) ;
//Perspective_map_al(Subs, &NewTransf, SrcBmap, AlphaMap,
//                  DstPoly,glModelColor);
//perspective_mask(hwnd,SubstBmap,SrcBmap,AlphaMap,SrcPoly,
//TESTS
//      DstPoly, glModelColor, InterpMode,MskBmap) ; //TESTS
bm_free(AlphaMap) ;
create_norm_bmap(hwnd,SrcBmap,&Normap,GIBound) ; //TESTS
if ( ToDisplay == 1 ) {
    InvalidateRect (hwnd, &GIBound, TRUE) ;
}

return 1 ;

```

}

```

int PASCAL WriteRgb (HWND hwnd,char *szFileName,MYBITMAP *Bmap)
{
    PHDR phdr;
    DWORD i,Offset ;
    long int Size ;
    HFILE hFile ;
    OFSTRUCT lpOpenBuff ;
    UINT ToWrite ;

    phdr.cols = Bmap->cols ;
    phdr.rows = Bmap->rows ;
    phdr.bp = 24 ;

    Size = (DWORD)phdr.cols*(DWORD)phdr.rows*(DWORD)3 ;
    hFile = OpenFile(szFileName,&lpOpenBuff,OF_CREATE | OF_WRITE) ;
    _lwrite(hFile,&phdr,sizeof(phdr)) ;

    Offset = 0 ;
    while ( Size > 0 ) {
        ToWrite = min(32768ul,Size) ;
        _lwrite(hFile,Bmap->gpic+Offset,ToWrite) ;
        Offset += ToWrite ;
        Size -= ToWrite ;
    }
    _lclose(hFile) ;

    return 1 ;
}

```

```

int PASCAL WriteGreyRgb (HWND hwnd,char *szFileName,MYBITMAP
*Bmap)
{
    PHDR phdr;
    DWORD i,Offset ;
    long int Size ;
    HFILE hFile ;
    OFSTRUCT lpOpenBuff ;
    UINT ToWrite ;
    BYTE huge *RgbArr ;
    BYTE huge *GreyTmp ;
    BYTE huge *Tmp ;
    DWORD GreySize ;

    phdr.cols = Bmap->cols ;
    phdr.rows = Bmap->rows ;

```

```

phdr.bp = 24 ;

Size = (DWORD)phdr.cols*(DWORD)phdr.rows*(DWORD)3 ;
RgbArr = (BYTE huge*)GlobalAllocPtr(GMEM_MOVEABLE,Size) ;
hFile = OpenFile(szFileName,&lpOpenBuff,OF_CREATE | OF_WRITE) ;
_lwrite(hFile,&phdr,sizeof(phdr)) ;

Tmp = RgbArr ;
GreyTmp = Bmap->gpic ;
GreySize = (DWORD)phdr.cols*(DWORD)phdr.rows ;
for ( i = 0 ; i < GreySize ; i++ ) {
    *(Tmp++) = *(GreyTmp) ;
    *(Tmp++) = *(GreyTmp) ;
    *(Tmp++) = *(GreyTmp++) ;
}

Offset = 0 ;
while ( Size > 0 ) {
    ToWrite = min(32768ul,Size) ;
    _lwrite(hFile,RgbArr+Offset,ToWrite) ;
    Offset += ToWrite ;
    Size -= ToWrite ;
}
_lclose(hFile) ;
GlobalFreePtr(RgbArr) ;

return 1 ;
}

```

```

int PASCAL WriteSign (HWND hwnd,char *szFileName,SIGN *Sign)
{
    DWORD i,Number ;
    HFILE hFile ;
    OFSTRUCT lpOpenBuff ;
    char String[100] ;

    Number = Sign->TrPoints.NumOfPoints ;
    sprintf(String,"%05d\n",10*Number+9+11) ;
    hFile = OpenFile(szFileName,&lpOpenBuff,OF_CREATE | OF_WRITE) ;
    _lwrite(hFile,String,strlen(String)) ;
    _lwrite(hFile,"orig.rgb\n",9) ;

    sprintf(String,"%05d %05d %05d %05d %05d %05d %05d %05d\n",
        OrgX[0],OrgY[0],OrgX[1],OrgY[1],OrgX[2],OrgY[2],OrgX[3],OrgY[3]) ;
    _lwrite(hFile,String,strlen(String)) ;

    sprintf(String,"%05d\n",Number) ;

```

```

    _lwrite(hFile,String,strlen(String));
    for ( i = 0 ; i < Number ; i++ ) {
        sprintf(String,"%4d %4d\n",Sign->TrPoints.TrackP[i].x,
                Sign->TrPoints.TrackP[i].y);
        _lwrite(hFile,String,strlen(String));
    }
    _lclose(hFile);

    return 1;
}

int PASCAL ReadSign(HWND hwnd,char *szFileName,SIGN *Sign)
{
    DWORD i,Number;
    HFILE hFile;
    OFSTRUCT lpOpenBuff;
    char String[100];
    int NumOfBytes;
    int NumOfPoints;
    int Xval,Yval;
    int Width,Height;
    BYTE huge *Ptr;

    hFile = OpenFile(szFileName,&lpOpenBuff, OF_READ);
    _lread(hFile,String,6); // Size of bytes in File.
    String[5] = '\0';
    sscanf(String,"%d",&NumOfBytes);
    _lread(hFile,String,9); // RGB File Name.
    String[8] = '\0';
    Ptr = ReadRGB(hwnd,String,&Width,&Height);
    bm_free(Sign->Bmap);
    Sign->Bmap = bm_alloc(Width,Height,glModelColor);
    Sign->Bmap->gpic = Ptr;
    create_lum_bmap(Sign->Bmap, &(Sign->BmapGrey));
    //WriteGreyRgb (hwnd,"greyorg.rgb",Sign->BmapGrey);

    _lread(hFile,String,48);
    sscanf(String,"%d %d %d %d %d %d %d %d",&(SignX[0]),&(SignY[0]),
        &(SignX[1]),&(SignY[1]),&(SignX[2]),&(SignY[2]),
        &(SignX[3]),&(SignY[3]));

    _lread(hFile,String,6); // Number of Points.
    String[5] = '\0';
    sscanf(String,"%d",&NumOfPoints);

    for ( i = 0 ; i < NumOfPoints ; i++ ) {
        _lread(hFile,String,10);
        String[9] = '\0';

```

```

        sscanf(String,"%d %d",&Xval,&Yval);
        Sign->TrPoints.TrackP[i].x = Xval;
        Sign->TrPoints.TrackP[i].y = Yval;
    }
    Sign->TrPoints.NumOfPoints = NumOfPoints;
    _lclose(hFile);

    return 1;
}

int PASCAL create_norm_bmap(HWND hwnd,MYBITMAP *Bmap,MYBITMAP
**NormBmap,
                        RECT ToValidate)
{
    BYTE huge * TmpNorm;
    BYTE huge * Tmp;
    int Cols,Rows;
    DWORD Size;
    long i,j,k;
    int RowLimit;

    Cols = Bmap->cols;
    Rows = Bmap->rows;

    if ( ToValidate.top == 0 ) {
        if ( *NormBmap != NULL ) bm_free(*NormBmap);

        *NormBmap = bm_alloc(Cols,Rows,GREY_MODEL);
        Size = (DWORD)Cols*(DWORD)Rows;
        (*NormBmap)->gpic = (BYTE
huge*)GlobalAllocPtr(GMEM_MOVEABLE,Size);
    }

    RowLimit = min(Rows,ToValidate.bottom);
    i = ToValidate.top;
    Tmp = Bmap->gpic +(DWORD)i*(DWORD)Cols*(DWORD)3;
    TmpNorm = (*NormBmap)->gpic+ (DWORD)i*(DWORD)Cols;
    while ( i < RowLimit ) {
        for ( j = 0 ; j < Cols ; j++ ) {
            *(TmpNorm++) = (BYTE)(((double)*(Tmp++))* 0.299 +
            (double)*(Tmp++)* 0.587 + (double)*(Tmp++)* 0.114);
        }
        i++;
    }
    return 1;
}

```

```

int PASCAL create_mask_bmap(HWND hwnd, MYBITMAP
*GreyMdlBmap, RPOINT *SrcPoly,
RPOINT *DstPoly, RECT Bound, MYBITMAP
**MaskBmap)
{
MYBITMAP *BoundedBmap = NULL ;
MYBITMAP *DiffBmap = NULL;
int Tresh ;
char String[100];
DWORD FromTime, ToTime ;
DWORD Sum ;

//WriteGreyRgb (hwnd, "model.rgb", GreyMdlBmap) ;

create_grey_bounded_bitmap(Bound, Normap, &BoundedBmap) ; //Copy the
area to

// be changed into a
// separate bitmap.

//WriteGreyRgb (hwnd, "bound0.rgb", BoundedBmap) ;

perspective(hwnd, GreyMdlBmap, Normap, SrcPoly,
DstPoly, GREY_MODEL, &BasicTransf) ;
//perspective_near(hwnd, GreyMdlBmap, Normap, SrcPoly,
// DstPoly, GREY_MODEL, &BasicTransf) ;
// Perform perspective
// grey of the current sign
// with itself into NormBmap.

subtract_bitmaps(Bound, Normap, BoundedBmap, &DiffBmap) ; // Subtract
BoundedBmap
// from the transformed
// sign area in NormBmap.

sum_grey_bitmap_value(DiffBmap, &Sum) ;
//sprintf(String, "Sum is %ld", Sum) ;
//MessageBox (hwnd, String, szAppName, MB_ICONEXCLAMATION |
MB_OK) ;
//WriteGreyRgb (hwnd, "diff.rgb", DiffBmap) ;
insert_grey_bounded_bitmap(Bound, BoundedBmap, Normap) ; // retrieve
original
//picture into Normap.

median_filter_5(hwnd, DiffBmap) ; // Creates a median
// filter on Diffbmap.

WriteGreyRgb (hwnd, "diff1.rgb", DiffBmap) ;

```

```

    get_tresh_for_occ(DiffBmap,&Tresh);
                                // Gets the treshold
                                // for occlusion.

    //sprintf(String,"Tresh Value = %d",Tresh);
    //MessageBox (hwnd, String, szAppName, MB_ICONEXCLAMATION |
    MB_OK);

    get_mask_bitmap(hwnd,DiffBmap,Tresh,MaskBmap);
    WriteGreyRgb (hwnd,"diff2.rgb",*MaskBmap);

    bm_free(BoundedBmap);
    bm_free(DiffBmap);

    return 1;
}

int PASCAL SonySearch(HWND hwnd, int frame, int port)
{
    char str[6];
    int i, input;
    const int N_TRY = 5;
    char DataBuffer[128];
    char InputBuffer[128];
    int len;
    int ErrCode;
    char String[50];
    COMSTAT lpStat;

    sprintf(str,"%05d",frame);

    DataBuffer[0] = 0x43; // Search
    WriteComm(port,DataBuffer, 1);
    InputBuffer[0] = 0x00;
    while ( InputBuffer[0] != 0x0A ) {
        ReadComm(port,InputBuffer,1);
        ErrCode = GetCommError(port,&lpStat);
        //sprintf(String,"Error Code = %d",ErrCode);
        //MessageBox (hwnd, String, szAppName, MB_ICONEXCLAMATION |
        MB_OK);
    }

    DataBuffer[0] = 0x55; // Frame mode
    WriteComm(port,DataBuffer, 1);
    InputBuffer[0] = 0x00;
    while ( InputBuffer[0] != 0x0A ) {
        ReadComm(port,InputBuffer,1);
    }

    for(i = 0; i < 5; i++) {

```

a31

```

    DataBuffer[0] = str[i] ; // Integer
    WriteComm(port,DataBuffer, 1);
    InputBuffer[0] = 0x00 ;
    while ( InputBuffer[0] != 0x0A ) {
        ReadComm(port,InputBuffer,1) ;
    }
}

DataBuffer[0] = 0x40 ; // Enter
WriteComm(port,DataBuffer, 1);
InputBuffer[0] = 0x00 ;
while ( InputBuffer[0] != 0x0A ) {
    ReadComm(port,InputBuffer,1) ;
}

InputBuffer[0] = 0x00 ;
while ( InputBuffer[0] != 0x01 ) {
    ReadComm(port,InputBuffer,1) ;
}

return((int)(InputBuffer[0]));
}

int PASCAL SonyRecord(HWND hwnd, int port)
{
    char str[6];
    int i, input;
    const int N_TRY = 5;
    BYTE DataBuffer[128];
    char InputBuffer[128];
    int len ;
    char String[50] ;
    int ErrCode ;
    COMSTAT lpStat ;

    //sprintf(String,"Before Frame Mode");
    //MessageBox (hwnd, String, szAppName, MB_ICONEXCLAMATION |
    MB_OK);
    DataBuffer[0] = 0x55 ; // FRAME # MODE
    WriteComm(port,DataBuffer, 1);
    InputBuffer[0] = 0x00 ;
    while ( InputBuffer[0] != 0x0A ) {
        ReadComm(port,InputBuffer,1) ;
        //ErrCode = GetCommError(port,&lpStat) ;
        //sprintf(String,"Error Code = %d",ErrCode) ;
        //MessageBox (hwnd, String, szAppName, MB_ICONEXCLAMATION |
        MB_OK);
    }
}

```



```

//sprintf(String,"After Frame Mode");
//MessageBox (hwnd, String, szAppName, MB_ICONEXCLAMATION |
MB_OK);
DataBuffer[0] = 0xEF ; // BLANK SEARCH DISABLE
WriteComm(port,DataBuffer, 1);
InputBuffer[0] = 0x00 ;
while ( InputBuffer[0] != 0x0A ) {
    ReadComm(port,InputBuffer,1);
}
//sprintf(String,"After Blank enable Mode");
//MessageBox (hwnd, String, szAppName, MB_ICONEXCLAMATION |
MB_OK);

```

```

sprintf(str,"%05ld",RecFrame);
sprintf(String,"STR = ***%s***",str);

```

```

DataBuffer[0] = 0xE0 ; // REC StndBy
WriteComm(port,DataBuffer, 1);
InputBuffer[0] = 0x00 ;
while ( InputBuffer[0] != 0x0A ) {
    ReadComm(port,InputBuffer,1);
}
while ( InputBuffer[0] != 0x01 ) {
    ReadComm(port,InputBuffer,1);
}
//sprintf(String,"After Rec StandBy");
//MessageBox (hwnd, String, szAppName, MB_ICONEXCLAMATION |
MB_OK);

```

```

for(i = 0; i < 5; i++) {
    DataBuffer[0] = str[i]; // Integer
    WriteComm(port,DataBuffer, 1);
    InputBuffer[0] = 0x00 ;
    while ( InputBuffer[0] != 0x0A ) {
        ReadComm(port,InputBuffer,1);
    }
}

```

```

DataBuffer[0] = 0x40 ; // Enter
WriteComm(port,DataBuffer, 1);
InputBuffer[0] = 0x00 ;
while ( InputBuffer[0] != 0x0A ) {
    ReadComm(port,InputBuffer,1);
}

```

```

RecFrame++;
sprintf(str,"%05ld",RecFrame);
for(i = 0; i < 5; i++) {

```

```

    DataBuffer[0] = str[i] ; // Integer
    WriteComm(port,DataBuffer, 1);
    InputBuffer[0] = 0x00 ;
    while ( InputBuffer[0] != 0x0A ) {
        ReadComm(port,InputBuffer,1) ;
    }
}
DataBuffer[0] = 0x40 ; // Enter
WriteComm(port,DataBuffer, 1);
InputBuffer[0] = 0x00 ;
while ( InputBuffer[0] != 0x0A ) {
    ReadComm(port,InputBuffer,1) ;
}
DataBuffer[0] = 0x40 ; // Enter
WriteComm(port,DataBuffer, 1);
InputBuffer[0] = 0x00 ;
while ( InputBuffer[0] != 0x0A ) {
    ReadComm(port,InputBuffer,1) ;
}
InputBuffer[0] = 0x00 ;
while ( InputBuffer[0] != 0x01 ) {
    ReadComm(port,InputBuffer,1) ;
}
DataBuffer[0] = 0xE6 ; // Frame Rec Mode
WriteComm(port,DataBuffer, 1);
InputBuffer[0] = 0x00 ;
while ( InputBuffer[0] != 0x0A ) {
    ReadComm(port,InputBuffer,1) ;
}
DataBuffer[0] = 0xE9 ; // Record
WriteComm(port,DataBuffer, 1);
InputBuffer[0] = 0x00 ;
while ( InputBuffer[0] != 0x0A ) {
    ReadComm(port,InputBuffer,1) ;
}
InputBuffer[0] = 0x00 ;
while ( InputBuffer[0] != 0x01 ) {
    ReadComm(port,InputBuffer,1) ;
}
}

return((int)(InputBuffer[0]));
}

```

```
static char RgbFileName[50] ;
```

```

UINT FAR PASCAL _export TimerProc(HWND hwnd, UINT message,UINT
wParam,

```

```
LONG lParam)
```

```
284
```

```

{
    DWORD Sum ;
    int i ;

    switch ( TimeElapsed ) {
    case 0:
        KillTimer(hwnd,1) ;
        if ( PictCount >= ToFr ) {
            InSerieMode = 0 ;
            break ;
        }
        sprintf(RgbFileName,"pict%d.rgb",PictCount++) ;
        load_picture_from_file(hwnd,RgbFileName) ;
        TimeElapsed++ ;
        SetTimer(hwnd,1,40,lpfnTimeProc) ;
        break ;
    case 1:
        KillTimer(hwnd,1) ;
        //SendMessage(hwnd,WM_COMMAND,IDM_AUTO_TRACK,0) ;
        if ( DummyFrames < DYMMIES ) {
            DummyFrames++ ;
            TimeElapsed++ ;
            SetTimer(hwnd,1,40,lpfnTimeProc) ;
            break ;
        }
        if ( ReplaceFlag == 1 ) {
            change_sign_by_tracking(hwnd,0) ;
        }
        TimeElapsed++ ;
        SetTimer(hwnd,1,40,lpfnTimeProc) ;
        break ;
    case 2:
        KillTimer(hwnd,1) ;
        draw_field_to_screen(hwnd,SrcBmap,IsOddLine) ;
        //draw_to_screen(SrcBmap,0,SrcBmap->cols) ;
        TimeElapsed++ ;
        SetTimer(hwnd,1,40,lpfnTimeProc) ;
        break ;
    case 3:
        KillTimer(hwnd,1) ;
        if ( IsOddLine == 1 ) { //TTTTTT
            SonyRecord(hwnd,PortNumber) ; //TTTTTT
            //IsOddLine = 1 - IsOddLine ;
            //TimeElapsed = 0 ;
            //break ;
        } // TTTTTT
        IsOddLine = 1 - IsOddLine ;
        TimeElapsed++ ;
        SetTimer(hwnd,1,1000,lpfnTimeProc) ;
    }
}

```

```

        break ;
    case 4:
        TimeElapsed++ ;
        break ;
    default:
        TimeElapsed = 0 ;
        break ;
}

```

```

return 1 ;
}

```

```

UINT FAR PASCAL _export TimerProc(HWND hwnd, UINT message,UINT
wParam,

```

```

LONG lParam)

```

```

{
    DWORD Sum ;
    int i ;

```

```

    switch ( TimeElapsed ) {
    case 0:
        KillTimer(hwnd,1) ;
        //MessageBox (hwnd, "Before Search", szAppName,
        MB_ICONEXCLAMATION | MB_OK) ;
        SonySearch(hwnd, SearchFrame,PortNumber) ;
        SearchFrame++ ;
        //MessageBox (hwnd, "After Search", szAppName,
        MB_ICONEXCLAMATION | MB_OK) ;
        mfg_setvframe(RGB) ;
        mfg_dacmode(TRUE_24);
        TimeElapsed++ ;
        mfg_snap(CAMERA,PAGE1) ;
        SetTimer(hwnd,1,40,lpfnTimeProc) ;
        break ;
    case 1:
        KillTimer(hwnd,1) ;
        if ( PictCount >= ToFr ) {
            InSerieMode = 0 ;
            break ;
        }
        sprintf(RgbFileName,"pict%d.rgb",PictCount++);
        load_field_from_card(hwnd,0) ;
        WriteRgb (hwnd,RgbFileName,SrcBmap) ;
        sprintf(RgbFileName,"pict%d.rgb",PictCount++);
        load_field_from_card(hwnd,1) ;
        WriteRgb (hwnd,RgbFileName,SrcBmap) ;
        SetTimer(hwnd,1,40,lpfnTimeProc) ;
    }
}

```

```

        TimeElapsed++;
        break ;
    default:
        TimeElapsed = 0 ;
        break ;
}

```

```

return 1 ;
}
*/

```

```

int PASCAL load_field_from_card(HWND hwnd,int NumOfField)
{
    DWORD Size ;
    BYTE huge *RedBytes ;
    BYTE huge *GreenBytes ;
    BYTE huge *BlueBytes ;
    int j,k ;
    DWORD Ticks ;
    char String[50] ;

```

```

    Ticks = GetTickCount() ;
    //Size = (DWORD)572*(DWORD)768*(DWORD)3 ;
    Size = (DWORD)286*(DWORD)768*(DWORD)3 ;
    if ( SrcBmap != NULL ) bm_free(SrcBmap) ;
    SrcBmap = bm_alloc(768,286,gIModelColor);
    SrcBmap->gpic = (BYTE.huge *)GlobalAllocPtr(GMEM_MOVEABLE,Size);
    RedBytes = SrcBmap->gpic ;
    GreenBytes = SrcBmap->gpic+1 ;
    BlueBytes = SrcBmap->gpic+2 ;
    mfg_setgframe(R);
    for ( k = NumOfField ; k < 572 ;k++ ) {
        mfg_brhline(PAGE0,0,k++,768,rpixarr) ;
        for ( j = 0 ; j < 768 ;j++ ) {
            *RedBytes = rpixarr[j] ;
            RedBytes += 3 ;
        }
    }
    mfg_setgframe(G);
    for ( k = NumOfField ; k < 572 ;k++ ) {
        mfg_brhline(PAGE0,0,k++,768,rpixarr) ;
        for ( j = 0 ; j < 768 ;j++ ) {
            *GreenBytes = rpixarr[j] ;
            GreenBytes += 3 ;
        }
    }
    mfg_setgframe(B);
    for ( k = NumOfField ; k < 572 ;k++ ) {

```

```

    mfg_brhline(PAGE0,0,k++,768,rpixarr);
    for ( j = 0 ; j < 768 ; j++ ) {
        *BlueBytes = rpixarr[j];
        BlueBytes += 3;
    }
}

sprintf(String,"Time = %ld",GetTickCount() - Ticks);
MessageBox (hwnd, szAppName,String,MB_ICONEXCLAMATION |
MB_OK);

teach_rgb_pallette(hwnd);
FileType = FILE_IS_RGB;
GetClientRect(hwnd,&ClientRect);
create_norm_bmap(hwnd,SrcBmap,&Normmap,ClientRect);
return 1;
}

int PASCAL load_picture_from_card(HWND hwnd)
{
    DWORD Size;
    BYTE huge *RedBytes;
    BYTE huge *GreenBytes;
    BYTE huge *BlueBytes;
    int j,k;

    Size = (DWORD)572*(DWORD)768*(DWORD)3;
    if ( SrcBmap != NULL ) bm_free(SrcBmap);
    SrcBmap = bm_alloc(768,572,glModelColor);
    SrcBmap->gpic = (BYTE huge *)GlobalAllocPtr(GMEM_MOVEABLE,Size);
    RedBytes = SrcBmap->gpic;
    GreenBytes = SrcBmap->gpic+1;
    BlueBytes = SrcBmap->gpic+2;
    mfg_setgframe(R);
    for ( k = 0 ; k < 572 ; k++ ) {
        mfg_brhline(PAGE0,0,k,768,rpixarr);
        for ( j = 0 ; j < 768 ; j++ ) {
            *RedBytes = rpixarr[j];
            RedBytes += 3;
        }
    }
    mfg_setgframe(G);
    for ( k = 0 ; k < 572 ; k++ ) {
        mfg_brhline(PAGE0,0,k,768,rpixarr);
        for ( j = 0 ; j < 768 ; j++ ) {
            *GreenBytes = rpixarr[j];
            GreenBytes += 3;
        }
    }
}

```

```

mfg_setgframe(B);
for ( k = 0 ; k < 572 ; k++ ) {
    mfg_brhline(PAGE0,0,k,768,rpixarr);
    for ( j = 0 ; j < 768 ; j++ ) {
        *BlueBytes = rpixarr[j];
        BlueBytes += 3 ;
    }
}
//if ( InterpMode == IN_TRILINEAR )
//    build_pyramid(hwnd,SrcBmap);

teach_rgb_pallette(hwnd);
FileType = FILE_IS_RGB ;
GetClientRect(hwnd,&ClientRect);
create_norm_bmap(hwnd,SrcBmap,&Normap,ClientRect);
return 1 ;
}

int PASCAL load_picture_from_file(HWND hwnd,char *FileName)
{
    int ColorFactor ;
    int Width,Height ;
    DWORD SrcSize ;
    DWORD i ;

    if ( glModelColor == GREY_MODEL )      {
        ColorFactor = 1 ;
        lpGreyRgb = ReadRGB (hwnd,FileName,&Width,&Height) ;
        lpPtr = lpGreyRgb ;
    }
    if ( glModelColor == COLOR_MODEL ) {
        ColorFactor = 3 ;
        lpPtr = ReadRGB (hwnd,FileName,&Width,&Height) ;
    }
    IsRgb = 1 ;
    bm_free(SrcBmap) ;
    SrcBmap = bm_alloc(Width,Height, glModelColor);
    if (SrcBmap == NULL) {
        MessageBox (hwnd, szAppName,"bm_alloc
Failed!",MB_ICONEXCLAMATION | MB_OK) ;
        return 0 ;
    }
    SrcSize = (DWORD)Width*(DWORD)(Height)*(DWORD)(ColorFactor);
    SrcBmap->gpPic = lpPtr ;
    if ( SplitMode == 1 && SrcBmap != NULL) {
        split_bitmap(SrcBmap,SrcBmap->typ,30) ; // NEW CALL
    }
    GetClientRect(hwnd,&ClientRect);

```

239

SUBSTITUTE SHEET (RULE 26)

```

        create_norm_bmap(hwnd,SrcBmap,&Normap,ClientRect) ;
        return 1 ;
    }

int PASCAL draw_field_to_screen(HWND hwnd,MYBITMAP *Bmap,int
IsOdd)
{
    int i,j;
    BYTE huge *Ptr ;
    BYTE huge *PtrR ;
    BYTE huge *PtrG ;
    BYTE huge *PtrB ;
    int Cols,Rows ;

    IsOdd = 1- IsOdd ; // PATCH
    Cols = Bmap->cols ;
    Rows = (DWORD)(Bmap->rows)*(DWORD)2 ;
    Ptr = Bmap->gpics ;
    //Ptr += (DWORD)IsOdd*(DWORD)Cols*(DWORD)3 ;
    PtrR = Ptr ;
    PtrG = Ptr+1 ;
    PtrB = Ptr+2 ;
    mfg_setgframe(R);
    for ( i = IsOdd ; i < Rows ; i +=2 ) {
        for ( j = 0 ; j < Cols ; j++ ) {
            rpixarr[j] = *PtrR ;
            PtrR += 3 ;
        }
        //PtrR += (DWORD)Cols*(DWORD)3 ;
        mfg_bwhline(PAGE0,0,i,Cols,rpixarr) ;
    }
    mfg_setgframe(G);
    for ( i = IsOdd ; i < Rows ; i +=2 ) {
        for ( j = 0 ; j < Cols ; j++ ) {
            rpixarr[j] = *PtrG ;
            PtrG += 3 ;
        }
        //PtrG += (DWORD)Cols*(DWORD)3 ;
        mfg_bwhline(PAGE0,0,i,Cols,rpixarr) ;
    }
    mfg_setgframe(B);
    for ( i = IsOdd ; i < Rows ; i +=2 ) {
        for ( j = 0 ; j < Cols ; j++ ) {
            rpixarr[j] = *PtrB ;
            PtrB += 3 ;
        }
        //PtrB += (DWORD)Cols*(DWORD)3 ;
        mfg_bwhline(PAGE0,0,i,Cols,rpixarr) ;
    }
}

```

290



```

    return 1 ;
}

int PASCAL get_fitness_of_vertexes(HWND hwnd,RECT Bound,MYBITMAP
*Bmap,
                                RPOINT *DstPnts,DWORD *Sum)
{
    MYBITMAP *BoundedBmap=NULL ;
    MYBITMAP *DiffBmap =NULL ;
    RPOINT    SrcPnts[4] ;

    l_cp_int_arr_to_RPOINT(SrcPnts, SignX, SignY,4) ;

    create_grey_bounded_bitmap(Bound,Normap,&BoundedBmap) ;//Copy the
area to
                                // be changed into a
                                //separate bitmap.

    perspective(hwnd,Bmap,Normap,SrcPnts,
                DstPnts, GREY_MODEL, &BasicTransf) ;
                                // Perform perspective
                                // grey of the current sign
                                // with itself into NormBmap.
    subtract_bitmaps(Bound,Normap,BoundedBmap,&DiffBmap) ; // Subtract
BoundedBmap
                                // from the transformed
                                // sign area in NormBmap.

    sum_grey_bitmap_value(DiffBmap,Sum) ;

    bm_free(BoundedBmap) ;
    bm_free(DiffBmap) ;

    return 1 ;
}

int PASCAL replace_sign(HWND hwnd,RPOINT *Dest)
{
    MYBITMAP *AlphaMap ;

    build_alpha_map(&AlphaMap, SrcBmap, Dest) ;
    perspective_al(hwnd,SubstBmap,SrcBmap,AlphaMap,SrcPoly,
                Dest, glModelColor, &BasicTransf) ;

    bm_free(AlphaMap) ;
    create_norm_bmap(hwnd,SrcBmap,&Normap,GIBound) ;
    return 1 ;
}

```

```

int PASCAL smooth_values(double NewVal, double *OutVal, double *ValArr,
                          int Num, double *Wheight)
{
    int i;
    double SumW = 0.0;
    double Calc;

    for (i = 0; i < Num-1; i++) {
        ValArr[i] = ValArr[i+1];
    }
    ValArr[Num-1] = NewVal;
    Calc = 0;
    for (i = 0; i < Num; i++) {
        Calc += ValArr[i]*Wheight[i];
        SumW += Wheight[i];
    }

    *OutVal = Calc/SumW;
    return 1;
}

```

```

int PASCAL create_subs_sign(HWND hwnd)
{
    //Perspective_Transform TpCurr;
    Perspective_Transform BasicTransf;
    RECT Rectan;
    int    Cols, Rows;
    DWORD  Size;
    RPOINT SrcPnts[4];
    RPOINT CmPoly[4];

```

```

    //Quad2Quad(hwnd, SrcPoly, DstPoly, &TpCurr);

```

```

    l_find_bound_rect(DstPoly, &Rectan);
    Cols = Rectan.right-Rectan.left+1;
    Rows = Rectan.bottom-Rectan.top+1;
    l_quad_in_new_orign(CmPoly, DstPoly, Rectan.left, Rectan.top, 4);

```

```

    bm_free(Subs);
    Subs = bm_alloc(Cols, Rows, COLOR_MODEL);
    Size = (DWORD)Cols*(DWORD)Rows*(DWORD)3;
    Subs->gpic = (BYTE huge*)GlobalAllocPtr(GMEM_MOVEABLE, Size);

```

```

    perspective(hwnd, SubstBmap, Subs, SrcPoly,
                CmPoly, COLOR_MODEL, &BasicTransf);

```

292

```

    return 1 ;
}

int PASCAL fill_mask_rect(MYBITMAP *Bmap)
{
    int i,j,k,l ;
    int Cols,Rows ;
    int Count ;
    int Index = 0 ;

    Cols = Bmap->cols ;
    Rows = Bmap->rows ;
    Count = 0 ;
    for ( i= 0 ; i < Cols ; i++ ) {
        if ( BITMAP_PLACE(Bmap,0,i) == 0 ) Count++ ;
        else {
            if ( Count > 0 ) {
                OcclusionRects[Index].top = 0 ;
                OcclusionRects[Index].left = i - Count+1 ;
                OcclusionRects[Index].bottom = Rows ;
                OcclusionRects[Index].right = i /*+Count/3 */ ; // IF BUG i.
                Index++ ;
                Count = 0 ;
            }
        }
    }

    if ( Index == 0 && NumOfValidOccRects > 0 ) {
        OcclusionRects[0] = ValidOccRects[0] ;
        OcclusionRects[0].right = min(OcclusionRects[0].right+2,Cols-1) ;
        OcclusionRects[0].bottom = Rows ;
        for ( k = OcclusionRects[0].top ; k < OcclusionRects[0].bottom ; k++ ) {
            for ( l = OcclusionRects[0].left ; l < OcclusionRects[0].right ; l++ ) {
                BITMAP_PLACE(Bmap,k,l) = 0 ;
            }
        }
        Index = 1 ;
    }
    NumOfOccRects = Index ;

    return 1 ;
}

```

```

int PASCAL ValidateOccRects(MYBITMAP *Bmap)
{
    int i, j, k, l ;
    RECT Intersection ;
    int Index = 0 ;

```

```

for ( j = 0 ; j < NumOfValidOccRects ; j++ ) {
    for ( i = 0 ; i < NumOfOccRects ; i++ ) {
        IntersectRect(&Intersection,&(OcclusionRects[i]),&(ValidOccRects[j])) ;
        if ( Intersection.bottom == 0 ) {
            for ( k = OcclusionRects[i].top ; k < OcclusionRects[i].bottom ; k++ )
            {
                for ( l = OcclusionRects[i].left ; l < OcclusionRects[i].right ; l++ ) {
                    BITMAP_PLACE(Bmap,k,l) = 1 ;
                }
            }
        }
    }
}
return 1 ;
}

int PASCAL copy_into_valid_rects()
{
    int i ;

    for ( i = 0 ; i < NumOfOccRects ; i++ ) {
        ValidOccRects[i] = OcclusionRects[i] ;
    }
    NumOfValidOccRects = NumOfOccRects ;
    return 1 ;
}

int PASCAL enlarge_area_of_noise(MYBITMAP *From,MYBITMAP *Bmap)
{
    int i, j ;
    int Cols,Rows ;
    RECT Rectan ;

    Cols = Bmap->cols ;
    Rows = Bmap->rows ;
    for ( j = 4 ; j < Cols-4 ; j+=3 ) {
        for ( i = 4 ; i < Rows-4 ; i++ ) {
            Rectan.top = i-3 ;
            Rectan.left = j-3 ;
            Rectan.right = j+3 ;
            Rectan.bottom = i+3 ;
            filter_noises_by_rects(From,Rectan,8,25,Bmap) ;
        }
    }
    return 1 ;
}

```

```
#include <windows.h>
#include <windowsx.h>
#include <commdlg.h>
#include <stdlib.h>
#include <math.h>
#include "const.h"
#include "bitmap.h"
#include "lines.h"
#include "track.h"
#include "min_mag.h"
#include "lib.h"
```

RECT Screen ;

MYBITMAP FAR \*dMaps[1 + DLVLS];

MYBITMAP FAR \*iMaps[1 + ILVLS];

int PASCAL refine\_alpha\_edges(MYBITMAP\* );

int PASCAL get\_longest\_occlusion(MYBITMAP \*,int ,int );

MYBITMAP FAR \*bm\_alloc(int xdim, int ydim, BMTYPE typ)

{  
MYBITMAP FAR\* bm;

bm = (MYBITMAP FAR \*)

GlobalAllocPtr(GMEM\_MOVEABLE, sizeof(MYBITMAP));

if (bm == NULL )

return NULL ;

bm->typ = typ;

bm->cols = xdim;

bm->rows = ydim;

bm->gpic = NULL ;

return (bm);

}

```
double PASCAL bilinear(double xs, double ys, MYBITMAP FAR* bmap)
```

```
{
    int      yi = (int) ys;
    double    dy = ys - (double) yi;
    int      xi = (int) xs;
    double    dx = xs - (double) xi;
    double    g00, g01, g10, g11, g1, g2;
```

```
    if (xi < 0 || xi >= bmap->cols)
```

```
        return (-1.0);
```

```
    if (yi < 0 || yi >= bmap->rows)
```

```
        return (-1.0);
```

```
    g00 = BITMAP_PLACE(bmap, yi, xi);
```

```
    g01 = (xi == bmap->cols - 1 ? g00 : BITMAP_PLACE(bmap, yi, xi+1));
```

```
    g10 = (yi == bmap->rows - 1 ? g00 : BITMAP_PLACE(bmap, yi+1, xi));
```

```
    g11 = (xi == bmap->cols - 1 || yi == bmap->rows - 1 ?
```

```
        g00 : BITMAP_PLACE(bmap, yi+1, xi+1));
```

```
    g1 = g00 + (g01-g00)*dx;
```

```
    g2 = g10 + (g11-g10)*dx;
```

```
    return(g1 + (g2-g1)*dy);
```

```
}
```

```
double PASCAL bilinear_red(double xs, double ys, MYBITMAP FAR* bmap)
```

```
{
    int      yi = (int) ys;
    double    dy = ys - (double) yi;
    int      xi = (int) xs;
    double    dx = xs - (double) xi;
    double    g00, g01, g10, g11, g1, g2;
```

```
    if (xi < 0 || xi >= bmap->cols)
```

```
        return (-1.0);
```

```
    if (yi < 0 || yi >= bmap->rows)
```

```
        return (-1.0);
```

```
    g00 = BITMAP_RGB_PLACE(bmap, yi, xi);
```

296

```

    g01 = (xi == bmap->cols - 1 ? g00 :
*(BITMAP_RGB_PLACE_PTR(bmap,yi,xi+1)));
    g10 = (yi == bmap->rows - 1 ? g00
*(BITMAP_RGB_PLACE_PTR(bmap,yi+1,xi)));
    g11 = (xi == bmap->cols - 1 || yi == bmap->rows - 1 ?
    g00 : *(BITMAP_RGB_PLACE_PTR(bmap,yi+1,xi+1)));
    g1 = g00 * (1.0 - dx) + g01 * dx;
    g2 = g10 * (1.0 - dx) + g11 * dx;
    return (g1 * (1.0 - dy) + g2 * dy);
}

```

double PASCAL bilinear\_green(double xs, double ys, MYBITMAP FAR\*  
bmap)

```

{
    int          yi = (int) ys;
    double        dy = ys - (double) yi;
    int          xi = (int) xs;
    double        dx = xs - (double) xi;
    double        g00, g01, g10, g11, g1, g2;

    if (xi < 0 || xi >= bmap->cols)
        return (-1.0);
    if (yi < 0 || yi >= bmap->rows)
        return (-1.0);
    g00 = *(BITMAP_RGB_PLACE_PTR(bmap,yi,xi)+1);
    g01 = (xi == bmap->cols - 1 ? g00 :
*(BITMAP_RGB_PLACE_PTR(bmap,yi,xi+1)+1));
    g10 = (yi == bmap->rows - 1 ? g00 :
*(BITMAP_RGB_PLACE_PTR(bmap,yi+1,xi)+1));
    g11 = (xi == bmap->cols - 1 || yi == bmap->rows - 1 ?
    g00 : *(BITMAP_RGB_PLACE_PTR(bmap,yi+1,xi+1)+1));
    g1 = g00 * (1.0 - dx) + g01 * dx;
    g2 = g10 * (1.0 - dx) + g11 * dx;

    return (g1 * (1.0 - dy) + g2 * dy);
}

```

```

double PASCAL bilinear_blue(double xs, double ys, MYBITMAP FAR* bmap)
{
    int      yi = (int) ys;
    double   dy = ys - (double) yi;
    int      xi = (int) xs;
    double   dx = xs - (double) xi;
    double   g00, g01, g10, g11, g1, g2;

    if (xi < 0 || xi >= bmap->cols)
        return (-1.0);
    if (yi < 0 || yi >= bmap->rows)
        return (-1.0);
    g00 = *(BITMAP_RGB_PLACE_PTR(bmap,yi,xi)+2);
    g01 = (xi == bmap->cols - 1 ? g00
    :*(BITMAP_RGB_PLACE_PTR(bmap,yi,xi+1)+2));
    g10 = (yi == bmap->rows - 1 ? g00
    :*(BITMAP_RGB_PLACE_PTR(bmap,yi+1,xi)+2));
    g11 = (xi == bmap->cols - 1 || yi == bmap->rows - 1 ?
        g00 : *(BITMAP_RGB_PLACE_PTR(bmap,yi+1,xi+1)+2));
    g1 = g00 * (1.0 - dx) + g01 * dx;
    g2 = g10 * (1.0 - dx) + g11 * dx;

    return (g1 * (1.0 - dy) + g2 * dy);
}

```

```

int PASCAL multmat(HWND hwnd, int Ma, int Na, int Nb, double *A, double
*B, double *C)
{
    int      i, j, k;

    for (i = 0; i < Ma; i++) {
        for (j = 0; j < Nb; j++) {
            C[(DWORD)i * (DWORD)Nb + (DWORD)j] = 0.0;
            for (k = 0; k < Na; k++) {
                C[(DWORD)i * (DWORD)Nb + (DWORD)j] +=
                    A[(DWORD)i * (DWORD)Na + (DWORD)k] *
                    B[(DWORD)k * (DWORD)Nb + (DWORD)j];
            }
        }
    }
}

```



```

    }
}
return 1 ;
}

```

```

double PASCAL resample_trilinear(MYBITMAP FAR* bmap, double xs,
double ys,

```

```

double D,int ColorModel)

```

```

{
int id,i,mag ;
double          fMag,hMag ;
double          g1,g2,alpha,g ;

if (D <= 1.0) // magnified pyramid level not used
    return (bilinear(xs, ys, bmap));
id = (int) ceil(D);
for (i = 1; i <= DLVLS; i++) {
    mag = (1 << i);
    if (mag >= id)
        break;
}

```

```

if (i > DLVLS) // interpolation is bi-linear
    return (bilinear(xs, ys, dMaps[ILVLS]));

```

```

fMag = mag;
hMag = mag / 2;

```

```

g1 = bilinear(xs / hMag, ys / hMag, dMaps[i - 1]);
g2 = bilinear(xs / fMag, ys / fMag, dMaps[i]);
alpha = (D - hMag) / hMag;
g = g1 * (1.0 - alpha) + g2 * alpha;

```

```

return (g);
}

```

```

double PASCAL resample_trilinear_red(MYBITMAP FAR* bmap, double xs,
double ys,

```

```

                                double D,int ColorModel)

{
int id,i,mag ;
double          fMag,hMag ;
double          g1,g2,alpha,g ;

    if (D <= 1.0) // magnified pyramid level not used
        return (bilinear_red(xs, ys, bmap));
    id = (int) ceil(D);
    for (i = 1; i <= DLVLS; i++) {
        mag = (1 << i);
        if (mag >= id)
            break;
    }

    if (i > DLVLS) // interpolation is bi-linear
        return (bilinear_red(xs, ys, dMaps[ILVLS]));

    fMag = mag;
    hMag = mag / 2;

    g1 = bilinear_red(xs / hMag, ys / hMag, dMaps[i - 1]);
    g2 = bilinear_red(xs / fMag, ys / fMag, dMaps[i]);
    alpha = (D - hMag) / hMag;
    g = g1 * (1.0 - alpha) + g2 * alpha;

    return (g);
}

double PASCAL resample_trilinear_green(MYBITMAP FAR* bmap, double
xs, double ys,
                                double D,int ColorModel)

{
int id,i,mag ;
double          fMag,hMag ;
double          g1,g2,alpha,g ;

    if (D <= 1.0) // magnified pyramid level not used

```

```

    return (bilinear_green(xs, ys, bmap));
    id = (int) ceil(D);
    for (i = 1; i <= DLVLS; i++) {
        mag = (1 << i);
        if (mag >= id)
            break;
    }

    if (i > DLVLS) // interpolation is bi-linear
        return (bilinear_green(xs, ys, dMaps[DLVLS]));

    fMag = mag;
    hMag = mag / 2;

    g1 = bilinear_green(xs / hMag, ys / hMag, dMaps[i - 1]);
    g2 = bilinear_green(xs / fMag, ys / fMag, dMaps[i]);
    alpha = (D - hMag) / hMag;
    g = g1 * (1.0 - alpha) + g2 * alpha;

    return (g);
}

double PASCAL resample_trilinear_blue(MYBITMAP FAR* bmap, double xs,
double ys,
double D, int ColorModel)
{
    int id, i, mag;
    double fMag, hMag;
    double g1, g2, alpha, g;

    if (D <= 1.0) // magnified pyramid level not used
        return (bilinear_blue(xs, ys, bmap));
    id = (int) ceil(D);
    for (i = 1; i <= DLVLS; i++) {
        mag = (1 << i);
        if (mag >= id)
            break;
    }

```

```

if (i > DLVLS) // interpolation is bi-linear
    return (bilinear_blue(xs, ys, dMaps[ILVLS]));

```

```

fMag = mag;
hMag = mag / 2;

```

```

g1 = bilinear_blue(xs / hMag, ys / hMag, dMaps[i - 1]);
g2 = bilinear_blue(xs / fMag, ys / fMag, dMaps[i]);
alpha = (D - hMag) / hMag;
g = g1 * (1.0 - alpha) + g2 * alpha;

```

```

return (g);
}

```

```

int PASCAL build_pyramid(HWND hwnd, MYBITMAP FAR * SrcBmap)
{
    int i;

```

```

    dMaps[0] = iMaps[0] = SrcBmap;
    for (i = 1; i <= DLVLS; i++) {
        dMaps[i] = minify(hwnd, dMaps[i - 1], 2);
    }
    return 1;
}

```

```

int PASCAL bm_free(MYBITMAP FAR* BtMap)
{
    if ( BtMap != NULL ) {
        GlobalFreePtr(BtMap->gpPic);
        GlobalFreePtr(BtMap);
    }
    return 1;
}

```

```

int PASCAL find_horiz_line(MYBITMAP *Bmap, POINT Point1, POINT Point2,
RPOINT *Line)
{

```

```

int i,j,Ri,Rj;
double Luma1[7][7];
double Luma2[7][7];
double r,g,b;
double Tresh = 70;
RPOINT Found1,Found2;
int Found;

for ( i = Point1.y-3, Ri = 0; Ri < 7; i++,Ri++) {
    for ( j = Point1.x-3, Rj = 0; Rj < 7; j++,Rj++) {
        r = *(BITMAP_RGB_PLACE_PTR(Bmap,i,j));
        g = *(BITMAP_RGB_PLACE_PTR(Bmap,i,j)+1);
        b = *(BITMAP_RGB_PLACE_PTR(Bmap,i,j)+2);
        Luma1[Ri][Rj] = r * 0.299 + g * 0.587 + b * 0.114;
    }
}

for ( i = Point2.y-3, Ri = 0; Ri < 7; i++,Ri++) {
    for ( j = Point2.x-3, Rj = 0; Rj < 7; j++,Rj++) {
        r = *(BITMAP_RGB_PLACE_PTR(Bmap,i,j));
        g = *(BITMAP_RGB_PLACE_PTR(Bmap,i,j)+1);
        b = *(BITMAP_RGB_PLACE_PTR(Bmap,i,j)+2);
        Luma2[Ri][Rj] = r * 0.299 + g * 0.587 + b * 0.114;
    }
}

Found = 0;
Found1.x = Point1.x;
Found1.y = Point1.y;
for ( i = 1; i < 6 && Found == 0; i++) {
    for ( j = 0; j < 6; j++) {
        if ( fabs(Luma1[i-1][j]-Luma1[i][j]) >= Tresh ) {
            if ( fabs(Luma1[i-1][j+1] - Luma1[i][j+1]) >= Tresh ||
                fabs(Luma1[i+1][j+1] - Luma1[i][j+1]) >= Tresh ) {
                Found1.x = Point1.x -3+j;
                Found1.y = Point1.y -3+i;
                Found = 1;
                break;
            }
        }
    }
}

```

```

    }
}
Found = 0;
Found2.x = Point2.x;
Found2.y = Point2.y;
for ( i = 1; i < 6 && Found == 0; i++ ) {
    for ( j = 0; j < 6; j++ ) {
        if ( fabs(Luma2[i-1][j]-Luma2[i][j]) >= Tresh ) {
            if ( fabs(Luma2[i-1][j+1] - Luma2[i][j+1]) >= Tresh ||
                fabs(Luma2[i+1][j+1] - Luma2[i][j+1]) >= Tresh ) {
                Found2.x = Point2.x -3+j;
                Found2.y = Point2.y -3+i;
                Found = 1;
                break;
            }
        }
    }
}
}
}

Line->y = (Found2.x*Found1.y-Found2.y*Found1.x)/(Found2.x-Found1.x);
Line->x = (Found1.y-Line->y)/Found1.x;

return 1;
}

```

```

int PASCAL find_vertic_line(MYBITMAP *Bmap, POINT Point1, POINT
Point2, RPOINT *Line)
{
    int i,j,Ri,Rj;
    double Luma1[7][7];
    double Luma2[7][7];
    double r,g,b;
    double Tresh = 70;
    RPOINT Found1,Found2;
    int Found;

```

```

    for ( i = Point1.y-3, Ri = 0; Ri < 7; i++,Ri++) {
        for ( j = Point1.x-3, Rj = 0; Rj < 7; j++,Rj++) {

```

```

        r = *(BITMAP_RGB_PLACE_PTR(Bmap,i,j)) ;
        g = *(BITMAP_RGB_PLACE_PTR(Bmap,i,j)+1) ;
        b = *(BITMAP_RGB_PLACE_PTR(Bmap,i,j)+2) ;
        Luma1[Ri][Rj] = r * 0.299 + g * 0.587 + b * 0.114;
    }
}
for ( i = Point2.y-3, Ri = 0 ; Ri < 7 ; i++, Ri++) {
    for ( j = Point2.x-3, Rj = 0 ; Rj < 7 ; j++, Rj++) {
        r = *(BITMAP_RGB_PLACE_PTR(Bmap,i,j)) ;
        g = *(BITMAP_RGB_PLACE_PTR(Bmap,i,j)+1) ;
        b = *(BITMAP_RGB_PLACE_PTR(Bmap,i,j)+2) ;
        Luma2[Ri][Rj] = r * 0.299 + g * 0.587 + b * 0.114;
    }
}
Found = 0 ;
Found1.x = Point1.x ;
Found1.y = Point1.y ;
for ( i = 0 ; i < 6 && Found == 0; i++ ) {
    for ( j = 0 ; j < 6 ; j++ ) {
        if ( fabs(Luma1[i][j]-Luma1[i][j+1]) >= Tresh ) {
            if ( fabs(Luma1[i+1][j] - Luma1[i+1][j+1]) >= Tresh ) {
                Found1.x = Point1.x -3+j ;
                Found1.y = Point1.y -3+i ;
                Found = 1 ;
                break ;
            }
        }
    }
}
Found = 0 ;
Found2.x = Point2.x ;
Found2.y = Point2.y ;
for ( i = 0 ; i < 6 && Found == 0 ; i++ ) {
    for ( j = 0 ; j < 6 ; j++ ) {
        if ( fabs(Luma2[i][j]-Luma2[i][j+1]) >= Tresh ) {
            if ( fabs(Luma2[i+1][j] - Luma2[i+1][j+1]) >= Tresh ) {
                Found2.x = Point2.x -3+j ;
                Found2.y = Point2.y -3+i ;
            }
        }
    }
}

```

```

        Found = 1;
        break;
    }
}
}
}

if ( Found2.x == Found1.x ) {
    Line->y = 0.0 ;
    Line->x = Found1.x;
} else {
    Line->y = (Found2.x*Found1.y-Found2.y*Found1.x)/(Found2.x-Found1.x) ;
    Line->x = (Found1.y-Line->y)/Found1.x ;
}

return 1 ;
}

```

```

#define HSx 5 /* Half search area (x) */
#define HSy 5 /* Half search area (y) */
#define HWx 4 /* Half correlation window (x) */
#define HWy 4 /* Half correlation window (y) */

```

```

int PASCAL create_lum_bmap(MYBITMAP *Bmap, MYBITMAP **LumBmap)
{
    BYTE huge * Tmp;
    BYTE huge * TmpB ;
    int Cols, Rows ;
    DWORD Size ;
    DWORD i ;

    Cols = Bmap->cols ;
    Rows = Bmap->rows ;

    Tmp = Bmap->gpics ;

```



```
bm_free(*LumBmap);
```

```
*LumBmap = bm_alloc(Cols,Rows,GREY_MODEL);
Size = (DWORD)Cols*(DWORD)Rows;
(*LumBmap)->gpic = (BYTE huge*)GlobalAllocPtr(GMEM_MOVEABLE,Size)
;
```

```
TmpB = (*LumBmap)->gpic;
Tmp = Bmap->gpic;
```

```
for ( i = 0 ; i < Size ; i++ ) {
    *(TmpB++) = (BYTE)((double)*(Tmp++)* 0.299 +
        (double)*(Tmp++)* 0.587 + (double)*(Tmp++)* 0.114);
}
return 1;
}
```

```
int PASCAL duplicate_bmap(MYBITMAP *FrBmap, MYBITMAP **ToBmap,int
Type)
```

```
{
    BYTE huge * Tmp;
    BYTE huge * TmpB;
    int Cols, Rows;
    DWORD Size;
    DWORD i;
    int ColorFactor=1;
```

```
    Cols = FrBmap->cols;
    Rows = FrBmap->rows;
    if ( Type == COLOR_MODEL ) ColorFactor = 3;
```

```
    Tmp = FrBmap->gpic;
```

```
    if ( *ToBmap != NULL ) bm_free(*ToBmap);
```

```
*ToBmap = bm_alloc(Cols,Rows,Type);
Size = (DWORD)Cols*(DWORD)Rows*(DWORD)ColorFactor;
```

```
(*ToBmap)->gpic = (BYTE huge*)GlobalAllocPtr(GMEM_MOVEABLE,Size) ;
```

```
TmpB = (*ToBmap)->gpic ;
```

```
Tmp = FrBmap->gpic ;
```

```
for ( i = 0 ; i < Size ; i++ ) {
    *(TmpB++) = *(Tmp++) ;
}
```

```
return 1 ;
}
```

```
int PASCAL create_grey_bounded_bitmap(RECT area, MYBITMAP
*Bmap,MYBITMAP **BoundBmap)
```

```
{
    int i, j ;
    int Cols, Rows ;
    DWORD    Size ;
    BYTE huge* Tmp ;
```

```
    Cols = area.right - area.left ;
    Rows = area.bottom - area.top ;
    bm_free(*BoundBmap) ;
    *BoundBmap = bm_alloc(Cols, Rows, GREY_MODEL) ;
    Size = (DWORD)Cols*(DWORD)Rows ;
    (*BoundBmap)->gpic = (BYTE
huge*)GlobalAllocPtr(GMEM_MOVEABLE,Size) ;
    Tmp = (*BoundBmap)->gpic ;
    for ( i = area.top ; i < area.bottom ; i++ ) {
        for ( j = area.left ; j < area.right ; j++ ) {
            *(Tmp++) = BITMAP_PLACE(Bmap,i,j) ;
        }
    }
}
```

```
return 1 ;
}
```

```

int PASCAL subtract_bitmaps(RECT area, MYBITMAP *From, MYBITMAP
*Subs,
                                MYBITMAP **Diff)
{
    DWORD Size ;
    int Cols, Rows ;
    int i, j ;
    BYTE huge* Tmp ;

    Cols = Subs->cols ;
    Rows = Subs->rows ;
    bm_free(*Diff) ;
    *Diff = bm_alloc(Cols, Rows, GREY_MODEL) ;
    Size = (DWORD)Cols*(DWORD)Rows ;
    (*Diff)->gpic = (BYTE huge*)GlobalAllocPtr(GMEM_MOVEABLE, Size) ;
    Tmp = (*Diff)->gpic ;
    for ( i = area.top ; i < area.bottom ; i++ ) {
        for ( j = area.left ; j < area.right ; j++ ) {
            *(Tmp++) = (BYTE)abs((int)(BITMAP_PLACE(From, i, j)-
                BITMAP_PLACE(Subs, i-area.top, j-area.left))) ;
        }
    }

    return 1 ;
}

int PASCAL insert_grey_bounded_bitmap(RECT area, MYBITMAP *ToInsert,
MYBITMAP *Into)
{
    int i, j ;

    for ( i = area.top ; i < area.bottom ; i++ ) {
        for ( j = area.left ; j < area.right ; j++ ) {
            BITMAP_PLACE(Into, i, j) =
                BITMAP_PLACE(ToInsert, i-area.top, j-area.left) ;
        }
    }
}

```

```

    return 1;
}

int PASCAL copy_grey_rect_from_frame(MYBITMAP *Into, MYBITMAP
*From,
                                RECT area)
{
    int i,j;

    for ( i = area.top ; i < area.bottom ; i++ ) {
        for ( j = area.left ; j < area.right ; j++ ) {
            BITMAP_PLACE(Into,i-area.top,j-area.left) = BITMAP_PLACE(From,i,j)
        }
    }

    return 1 ;
}

int PASCAL build_alpha_map(MYBITMAP **AlphaMap, MYBITMAP
*SrcBmap,RPOINT *Vertex)
{
    int i,j;
    RECT Rectan ;
    int Cols,Rows ;
    SEG lines[4] ;
    DWORD Size ;
    RPOINT Points[4] ;
    double Xval,Xmin,Xmax ;
    int Curr ;

    Cols = SrcBmap->cols ;
    Rows = SrcBmap->rows ;
    l_find_bound_rect(Vertex, &Rectan) ;
    for ( i = 0 ; i < 3 ; i++ ) {
        mksegment(Vertex[i].x, Vertex[i].y, Vertex[i+1].x, Vertex[i+1].y,&(lines[i])) ;
    }
}

```

```

mksegment(Vertex[3].x, Vertex[3].y, Vertex[0].x, Vertex[0].y, &(lines[3]));
*AlphaMap = bm_alloc(Cols, Rows, GREY_MODEL);
Size = (DWORD)Cols*(DWORD)Rows;
(*AlphaMap)->gpPic =
    (BYTE huge*)GlobalAllocPtr(GMEM_MOVEABLE |
    GMEM_ZEROINIT, Size);
for( i = Rectan.top; i < Rectan.bottom; i++ ) {
    Curr = 0;
    for( j = 0; j < 4; j++ ) {
        if( lines[j].a != 0.0 ) {
            Xval = -1.0 *(lines[j].b*(double)i + lines[j].c)/lines[j].a;
            if( Xval <= Cols && Xval >= 0 ) {
                if( i <= (int)(lines[j].ymax) && i >= (int)(lines[j].ymin) )
                    Points[Curr++].x = Xval;
            }
        }
    }
    Xmin = Points[0].x;
    Xmax = Xmin;
    for( j = 1; j < Curr; j++ ) {
        Xmin = min(Points[j].x, Xmin);
        Xmax = max(Points[j].x, Xmax);
    }
    for( j = (int)Xmin; j <= (int)Xmax; j++ ) {
        BITMAP_PLACE((*AlphaMap), i, j) = 1;
    }
}
refine_alpha_edges(*AlphaMap);
return 1;
}

int PASCAL split_bitmap(MYBITMAP *Bmap, int ColorModel, int FromRow)
{
    DWORD i, k;
    int Cols, Rows;
    DWORD Place;
    BYTE huge *Dup;
    int ColorFactor;

```

DWORD Offset,Size,Offset2 ;

```

ColorFactor = 3 ;
Cols = Bmap->cols ;
Rows = Bmap->rows ;
Size = (DWORD)Rows*(DWORD)Cols*(DWORD)ColorFactor ;
Rows = (Bmap->rows-FromRow) /2 ;
if ( ColorModel == GREY_MODEL ) ColorFactor = 1;
Offset = (DWORD)FromRow*(DWORD)Cols*(DWORD)ColorFactor ;
Place = (DWORD)Rows*(DWORD)Cols*(DWORD)ColorFactor ;
Dup = Bmap->gpic + Size - Offset*(DWORD)2 ;
Offset2 = Offset*(DWORD)2 ;
for ( i = 0 ; i < Offset2 ; i++ ) {
    *(Dup++) = *(Bmap->gpic +Place+i) ;
}
Dup = Bmap->gpic+Place ;
k = 0 ;
for ( i = Offset ; i < Place ; i++ ) {
    *(Dup+ k++) = *(Bmap->gpic +i) ;
}
Size = (DWORD)Cols*(DWORD)6*(DWORD)ColorFactor ; // 6 Rows of
Black.
for ( i = 0 ; i < Size ; i++ ) {
    *(Dup++) = 0 ;
}

return 1 ;
}

```

```

int PASCAL refine_alpha_edges(MYBITMAP* AlphaMap)
{
    int i,j;
    int Cols,Rows ;
    BYTE Far,Close ;
    int Count = 0 ;

```

```

    Cols = AlphaMap->cols ;
    Rows = AlphaMap->rows ;

```

```

Far = 170 ;
Close = 85 ;
for ( i = 2 ; i < Rows ; i++ ) {
    for ( j = 2 ; j < Cols-1 ; j++ ) {
        if ( BITMAP_PLACE(AlphaMap,i,j) == 1 ) {
            /*
            if ( BITMAP_PLACE(AlphaMap,i,j-1) == 0 ) {
                BITMAP_PLACE(AlphaMap,i,j-2) = Far ;
                BITMAP_PLACE(AlphaMap,i,j-1) = Close ;
            }
            if ( BITMAP_PLACE(AlphaMap,i,j+1) == 0 ) {
                BITMAP_PLACE(AlphaMap,i,j+2) = Far ;
                BITMAP_PLACE(AlphaMap,i,j+1) = Close ;
                j++ ;
            }
            */
            if ( BITMAP_PLACE(AlphaMap,i-1,j) == 0 ) {
                BITMAP_PLACE(AlphaMap,i,j) = Far ;
                BITMAP_PLACE(AlphaMap,i+1,j) = Close ;
            }
            if ( BITMAP_PLACE(AlphaMap,i+1,j) == 0 ) {
                BITMAP_PLACE(AlphaMap,i,j) = Far ;
                BITMAP_PLACE(AlphaMap,i-1,j) = Close ;
            }
        }
    }
}

/*
for ( i = 2 ; i < Rows ; i++ ) {
    for ( j = 2 ; j < Cols ; j++ ) {
        if ( BITMAP_PLACE(AlphaMap,i,j) > 2 &&
        BITMAP_PLACE(AlphaMap,i,j-1) == 0 ) {
            Array[Count].x = j-1 ;
            Array[Count++].y = i ;
        }
    }
}

```

```

for ( i = 0 ; i < Count-1 ; i++ ) {
    Len = Array[i].x-Array[i+1].x ;
    if ( Len == 0 ) continue ;
    Unit = 256/Len ;
    Line = Array[i].y ;
    for ( j = Array[i].x, k = 1 ; Len > 0 ; Len-- ) {
        BITMAP_PLACE(AlphaMap,Line,j) = (DWORD)Unit*(DWORD)k ;
        j-- ;
        k++ ;
    }
}
*/
return 1 ;
}

```

/\*

```

int PASCAL get_mask_bitmap(HWND hwnd,MYBITMAP *Bmap,int
Tresh,MYBITMAP **MaskBmap)
{
    int Cols, Rows ;
    BYTE huge *Tmp ;
    BYTE huge *BmapPtr ;
    DWORD Size ,indx ;
    int i,j,k ;
    int Limit,Count,Value ;
    MYBITMAP *ToBmap ;

```

```

    Cols = Bmap->cols ;
    Rows = Bmap->rows ;
    bm_free(*MaskBmap) ;
    *MaskBmap = bm_alloc(Cols,Rows,GREY_MODEL) ;
    Size = (DWORD)Cols*(DWORD)Rows ;
    (*MaskBmap)->gpic = (BYTE
huge*)GlobalAllocPtr(GMEM_MOVEABLE,Size) ;
    Tmp = (*MaskBmap)->gpic ;
    BmapPtr = Bmap->gpic ;
    for ( indx = 0 ; indx < Size ; indx++ ) {

```



```

    if ( *(BmapPtr++) <= Tresh ) {
        *(Tmp++) = 1 ;
    } else {
        *(Tmp++) = 0 ;
    }
}

```

```

duplicate_bmap(*MaskBmap, &ToBmap, GREY_MODEL);
enlarge_area_of_noise(ToBmap, *MaskBmap);
WriteGreyRgb (hwnd, "mask0.rgb", *MaskBmap);
bm_free(ToBmap);

```

```

Limit = Rows/3 ;
for ( j = 0 ; j < Cols ; j++ ) {    // 0 Means Occlusion. 1 Means OK.
    Count = get_longest_occlusion(*MaskBmap, j, Rows);
    if ( Count < Limit ) {
        Value = 1 ;    //
        //for ( i = 0 ; i < Rows ; i++ ) {
        //    BITMAP_PLACE((*MaskBmap), i, j) = 1 ;
        //}
    } else Value = 0 ;    //
    for ( i = 0 ; i < Rows ; i++ ) {    //
        BITMAP_PLACE((*MaskBmap), i, j) = Value ; //
    }    //
}

```

```

Count = 0 ;
for ( j = 0 ; j < Cols ; j++ ) {
    if ( BITMAP_PLACE((*MaskBmap), 0, j) == 0 ) Count++ ;
    else {
        if ( Count == 0 ) continue ;
        if ( Count < 6 ) {
            for ( k = j-Count ; Count > 0 ; Count--, k++ ) {
                for ( i = 0 ; i < Rows ; i++ ) {
                    BITMAP_PLACE((*MaskBmap), i, k) = 1 ;
                }
            }
        } else { // Count >= 6

```

```

for ( i = 0 ; i < Rows ; i++ ) {

    if ( j > (Count+12) ) Limit = j-(Count+12) ;
    else Limit = 0 ;
    BITMAP_PLACE((*MaskBmap),i,Limit) = 0 ;
    if ( j > (Count+11) ) Limit = j-(Count+11) ;
    else Limit = 0 ;
    BITMAP_PLACE((*MaskBmap),i,Limit) = 0 ;
    if ( j > (Count+10) ) Limit = j-(Count+10) ;
    else Limit = 0 ;
    BITMAP_PLACE((*MaskBmap),i,Limit) = 0 ;
    if ( j > (Count+9) ) Limit = j-(Count+9) ;
    else Limit = 0 ;
    BITMAP_PLACE((*MaskBmap),i,Limit) = 0 ;
    if ( j > (Count+8) ) Limit = j-(Count+8) ;
    else Limit = 0 ;
    BITMAP_PLACE((*MaskBmap),i,Limit) = 0 ;
    if ( j > (Count+7) ) Limit = j-(Count+7) ;
    else Limit = 0 ;
    BITMAP_PLACE((*MaskBmap),i,Limit) = 0 ;
    if ( j > (Count+6) ) Limit = j-(Count+6) ;
    else Limit = 0 ;
    BITMAP_PLACE((*MaskBmap),i,Limit) = 0 ;
    if ( j > (Count+5) ) Limit = j-(Count+5) ;
    else Limit = 0 ;
    BITMAP_PLACE((*MaskBmap),i,Limit) = 0 ;
    if ( j > (Count+4) ) Limit = j-(Count+4) ;
    else Limit = 0 ;
    BITMAP_PLACE((*MaskBmap),i,Limit) = 0 ;
    if ( j > (Count+3) ) Limit = j-(Count+3) ;
    else Limit = 0 ;
    BITMAP_PLACE((*MaskBmap),i,Limit) = 0 ;
    if ( j > (Count+2) ) Limit = j-(Count+2) ;
    else Limit = 0 ;
    BITMAP_PLACE((*MaskBmap),i,Limit) = 0 ;
    if ( j > (Count+1) ) Limit = j-(Count+1) ;
    else Limit = 0 ;
    BITMAP_PLACE((*MaskBmap),i,Limit) = 0 ;

```

```

        BITMAP_PLACE((*MaskBmap),i,j-Count) = 0;
        BITMAP_PLACE((*MaskBmap),i,j) = 0 ;
        Limit = min(j+1,Cols-1) ;
        BITMAP_PLACE((*MaskBmap),i,Limit) = 0 ;
        Limit = min(j+2,Cols-1) ;
        BITMAP_PLACE((*MaskBmap),i,Limit) = 0 ;
        Limit = min(j+3,Cols-1) ;
        BITMAP_PLACE((*MaskBmap),i,Limit) = 0;
        Limit = min(j+4,Cols-1) ;
        BITMAP_PLACE((*MaskBmap),i,Limit) = 0 ;
        Limit = min(j+5,Cols-1) ;
        BITMAP_PLACE((*MaskBmap),i,Limit) = 0 ;
        Limit = min(j+6,Cols-1) ;
        BITMAP_PLACE((*MaskBmap),i,Limit) = 0 ;
        Limit = min(j+7,Cols-1) ;
        BITMAP_PLACE((*MaskBmap),i,Limit) = 0 ;
        Limit = min(j+8,Cols-1) ;
        BITMAP_PLACE((*MaskBmap),i,Limit) = 0 ;
    }
    j = min(j+6,Cols-1) ;
    Count = 0 ;
}
}
}
return 1 ;
}
*/

int PASCAL get_mask_bitmap(HWND hwnd,MYBITMAP *Bmap,int
Tresh,MYBITMAP **MaskBmap)
{
    int Cols, Rows ;
    BYTE huge *Tmp ;
    BYTE huge *BmapPtr ;
    DWORD Size ,indx ;
    int i,j,k ;
    int Limit,Count,Value ;
    MYBITMAP *TempBmap ;

```

RECT Rectan ;

```

Cols = Bmap->cols ;
Rows = Bmap->rows ;
TempBmap = bm_alloc(Cols,Rows,GREY_MODEL) ;
Size = (DWORD)Cols*(DWORD)Rows ;
TempBmap->gpic = (BYTE huge*)GlobalAllocPtr(GMEM_MOVEABLE,Size) ;
Tmp = TempBmap->gpic ;
BmapPtr = Bmap->gpic ;
for ( indx = 0 ; indx < Size ; indx++ ) {
    if ( *(BmapPtr++) <= Tresh ) {
        *(Tmp++) = 1 ;
    } else {
        *(Tmp++) = 0 ;
    }
}

```

```

Limit = Rows/3 ;
for ( j = 0 ; j < Cols ; j++ ) { // 0 Means Occlusion. 1 Means OK.
    Count = get_longest_occlusion(TempBmap,j,Rows) ;
    if ( Count >= Limit ) {
        BITMAP_PLACE(TempBmap,0,j) = 0 ;
    } else {
        BITMAP_PLACE(TempBmap,0,j) = 1 ;
    }
}

```

```

Count = 0 ;
for ( j = 0 ; j < Cols ; j++ ) {
    if ( BITMAP_PLACE(TempBmap,0,j) == 0 ) Count++ ;
    else {
        if ( Count == 0 ) continue ;
        if ( Count < 6 ) {
            for ( k = j-Count ; Count > 0 ; Count--, k++ ) {
                for ( i = 0 ; i < Rows ; i++ ) {
                    BITMAP_PLACE(TempBmap,i,k) = 1 ;
                }
            }
        }
    }
}

```

```

    }
} else { // Count >= 6
    for ( k = j-Count ; Count > 0 ; Count--, k++ ) {
        for ( i = 0 ; i < Rows ; i++ ) {
            BITMAP_PLACE(TempBmap,i,k) = 0 ;
        }
    }
    for ( i = 0 ; i < Rows ; i++ ) {
        if ( j > (Count+5) ) Limit = j-(Count+5) ;
        else Limit = 0 ;
        BITMAP_PLACE(TempBmap,i,Limit) = 0 ;
        if ( j > (Count+4) ) Limit = j-(Count+4) ;
        else Limit = 0 ;
        BITMAP_PLACE(TempBmap,i,Limit) = 0 ;
        if ( j > (Count+3) ) Limit = j-(Count+3) ;
        else Limit = 0 ;
        BITMAP_PLACE(TempBmap,i,Limit) = 0 ;
        if ( j > (Count+2) ) Limit = j-(Count+2) ;
        else Limit = 0 ;
        BITMAP_PLACE(TempBmap,i,Limit) = 0 ;
        if ( j > (Count+1) ) Limit = j-(Count+1) ;
        else Limit = 0 ;
        BITMAP_PLACE(TempBmap,i,Limit) = 0 ;
        Limit = min(j+1,Cols-1) ;
        BITMAP_PLACE(TempBmap,i,Limit) = 0 ;
        Limit = min(j+2,Cols-1) ;
        BITMAP_PLACE(TempBmap,i,Limit) = 0 ;
        Limit = min(j+3,Cols-1) ;
        BITMAP_PLACE(TempBmap,i,Limit) = 0 ;
        Limit = min(j+4,Cols-1) ;
        BITMAP_PLACE(TempBmap,i,Limit) = 0 ;
        Limit = min(j+5,Cols-1) ;
        BITMAP_PLACE(TempBmap,i,Limit) = 0 ;
        j = min(j+5,Cols-1) ;
    }
}
Count = 0 ;
}

```

```

    }
    duplicate_bmap(TempBmap, MaskBmap, GREY_MODEL);
    //WriteGreyRgb (hwnd,"mask0.rgb",*MaskBmap);
    for ( i = 5 ; i < Rows-5 ; i++ ) {
        for ( j = 5 ; j < Cols-5 ; j++ ) {
            if ( BITMAP_PLACE(TempBmap,i,j) == 0 ) {
                Rectan.top = i-5 ;
                Rectan.left = j ;
                Rectan.bottom = i+5 ;
                Rectan.right = j+5 ;
                filter_noises_by_rects(TempBmap,Rectan,6,30,*MaskBmap) ;
                Rectan.left = j-5 ;
                Rectan.right = j ;
                filter_noises_by_rects(TempBmap,Rectan,6,30,*MaskBmap) ;
                Rectan.right = j+5 ;
                filter_noises_by_rects(TempBmap,Rectan,11,55,*MaskBmap) ;
            }
        }
    }
    bm_free(TempBmap);
    //WriteGreyRgb (hwnd,"mask1.rgb",*MaskBmap);
    //Limit = Rows/4 ;
    for ( j = 1 ; j < Cols-2 ; j++ ) { // 0 Means Occlusion. 1 Means OK.
        //Count = get_longest_occlusion(*MaskBmap,j,Rows);
        if ( BITMAP_PLACE(*MaskBmap,0,j-1) == 1 &&
            (BITMAP_PLACE(*MaskBmap,0,j+2) == 1 ||
             BITMAP_PLACE(*MaskBmap,0,j+1) == 1) ) {
            for ( i = 0 ; i < Rows ; i++ ) {
                BITMAP_PLACE(*MaskBmap,i,j) = 1 ;
            }
        }
    }
    //WriteGreyRgb (hwnd,"mask2.rgb",*MaskBmap);

    return 1 ;
}

```

```

int PASCAL get_longest_occlusion(MYBITMAP *Bmap,in: col,int rows)

```

320

**SUBSTITUTE SHEET (RULE 26)**

```

{
  int i ;
  int Count = 0 ;
  int Longest = 0 ;

  for ( i = 0 ; i < rows ; i++ ) {
    if ( BITMAP_PLACE(Bmap,i,col) == 0 ) Count++ ;
    else {
      if ( Count > Longest ) {
        Longest = Count ;
        Count = 0 ;
      }
    }
  }
  if ( Count > Longest ) {
    Longest = Count ;
  }
  return Longest ;
}

```

```

double PASCAL bilinear_rgb(double xs, double ys, MYBITMAP FAR* bmap,
                           double *Ptr)

```

```

{
  int      yi = (int) ys;
  double   dy = ys - (double) yi;
  int      xi = (int) xs;
  double   dx = xs - (double) xi;
  double   g00, g01, g10, g11, g1, g2;
  BYTE     huge *PtrX1 ;
  BYTE     huge *PtrY1 ;
  BYTE     huge *PtrX1Y1 ;
  BYTE     huge *PtrXY ;

  PtrXY = BITMAP_RGB_PLACE_PTR(bmap,yi,xi) ;
  PtrX1 = BITMAP_RGB_PLACE_PTR(bmap,yi,xi+1) ;
  PtrY1 = BITMAP_RGB_PLACE_PTR(bmap,yi+1,xi) ;
  PtrX1Y1 = BITMAP_RGB_PLACE_PTR(bmap,yi+1,xi+1) ;

```

```

    g00 = PtrXY[0] ;
    g01 = (xi == bmap->cols - 1 ? g00 : PtrX1[0]);
    g10 = (yi == bmap->rows - 1 ? g00 : PtrY1[0]);
    g11 = (xi == bmap->cols - 1 || yi == bmap->rows - 1 ? g00 : PtrX1Y1[0]);
    g1 = g00 * (1.0 - dx) + g01 * dx;
    g2 = g10 * (1.0 - dx) + g11 * dx;
    Ptr[0] = g1 * (1.0 - dy) + g2 * dy ;

    g00 = PtrXY[1] ;
    g01 = (xi == bmap->cols - 1 ? g00 : PtrX1[1]);
    g10 = (yi == bmap->rows - 1 ? g00 : PtrY1[1]);
    g11 = (xi == bmap->cols - 1 || yi == bmap->rows - 1 ? g00 : PtrX1Y1[1]);
    g1 = g00 * (1.0 - dx) + g01 * dx;
    g2 = g10 * (1.0 - dx) + g11 * dx;
    Ptr[1] = g1 * (1.0 - dy) + g2 * dy ;

    g00 = PtrXY[2] ;
    g01 = (xi == bmap->cols - 1 ? g00 : PtrX1[2]);
    g10 = (yi == bmap->rows - 1 ? g00 : PtrY1[2]);
    g11 = (xi == bmap->cols - 1 || yi == bmap->rows - 1 ? g00 : PtrX1Y1[2]);
    g1 = g00 * (1.0 - dx) + g01 * dx;
    g2 = g10 * (1.0 - dx) + g11 * dx;
    Ptr[2] = g1 * (1.0 - dy) + g2 * dy ;

    return (0.0);
}

int PASCAL split_bitmap_frame(MYBITMAP *SrcBmap, MYBITMAP **F1,
MYBITMAP **F2)
{
    int Rows, Cols, MidRows ;
    DWORD Size, Limit, indx ;
    int i, ColorFactor ;
    BYTE huge * CurrPtr;
    BYTE huge * Ptr;

    Rows = SrcBmap->rows ;
    Cols = SrcBmap->cols ;

```



```

    if ( SrcBmap->typ == COLOR_MODEL ) ColorFactor = 3 ;
    else ColorFactor = 1 ;
    if ( Rows %2 == 0 ) {
        MidRows = Rows/2 ;
    } else {
        MidRows = Rows/2 +1 ;
    }
    Size = (DWORD)MidRows*(DWORD)Cols*(DWORD)ColorFactor ;
    *F1 = bm_alloc(Cols,MidRows,SrcBmap->typ) ;
    (*F1)->gpPic = (BYTE huge*)GlobalAllocPtr(GMEM_MOVEABLE,Size) ;
    Limit = (DWORD)Cols*(DWORD)ColorFactor ;
    CurrPtr = (*F1)->gpPic ;
    for ( i = 0 ; i < Rows ; i+=2 ) {
        if ( ColorFactor == 1 ) {
            Ptr = BITMAP_PLACE_PTR(SrcBmap,i,0) ;
        } else {
            Ptr = BITMAP_RGB_PLACE_PTR(SrcBmap,i,0) ;
        }
        for ( indx = 0 ; indx < Limit ; indx++ ) {
            *(CurrPtr++) = *(Ptr++) ;
        }
    }
}

```

```

MidRows = Rows - MidRows ;
Size = (DWORD)MidRows*(DWORD)Cols*(DWORD)ColorFactor ;
*F2 = bm_alloc(Cols,MidRows,SrcBmap->typ) ;
(*F2)->gpPic = (BYTE huge*)GlobalAllocPtr(GMEM_MOVEABLE,Size) ;
CurrPtr = (*F2)->gpPic ;
for ( i = 1 ; i < Rows ; i+=2 ) {
    if ( ColorFactor == 1 ) {
        Ptr = BITMAP_PLACE_PTR(SrcBmap,i,0) ;
    } else {
        Ptr = BITMAP_RGB_PLACE_PTR(SrcBmap,i,0) ;
    }
    for ( indx = 0 ; indx < Limit ; indx++ ) {
        *(CurrPtr++) = *(Ptr++) ;
    }
}
}

```

```

    return 1 ;
}

int PASCAL sum_grey_bitmap_value(MYBITMAP *Bmap,DWORD *Sum)
{
    DWORD i;
    int Cols,Rows ;
    DWORD Size ;
    DWORD CurrSum ;
    BYTE huge *Ptr ;

    Cols = Bmap->cols ;
    Rows = Bmap->rows ;
    Ptr = Bmap->gpic ;

    Size = (DWORD)Cols*(DWORD)Rows ;
    CurrSum = 0 ;
    for ( i = 0 ; i < Size ; i++ ) {
        CurrSum += *(Ptr++) ;
    }
    *Sum = CurrSum ;

    return 1 ;
}

int PASCAL filter_noises_by_rects(MYBITMAP *Source,RECT Rectan,int
LowLimit,

                                int HighLimit,MYBITMAP *Target)
{
    int i,j ;
    int Count = 0 ;
    int Left,Right,Top,Bottom ;
    int Value ;

    Left = Rectan.left ;
    Top = Rectan.top ;
    Right = Rectan.right ;

```

```
Bottom = Rectan.bottom ;

for ( i = Top ; i < Bottom ; i++ ) {
    for ( j = Left ; j < Right ; j++ ) {
        if ( BITMAP_PLACE(Source,i,j) == 0 ) Count++ ;
    }
}
Value = -1 ;
if ( Count < LowLimit ) Value = 1 ;
if ( Count > HighLimit ) Value = 0 ;
if ( Value == -1 ) return 1 ;

for ( i = Top ; i < Bottom ; i++ ) {
    for ( j = Left ; j < Right ; j++ ) {
        BITMAP_PLACE(Target,i,j) = Value ;
    }
}

return 1 ;
}
```

```

/*
 *   int qrsolv(M,size,b)
 *
 *   Solves the linear system :  $Mx = b$  using the QR Decomposition.
 *   On output b is overwritten by the solution x.
 *   Algorithm :
 *   1)qrdecomp M into Q & R (coded in M,m1,m2).
 *   2)multiply b by Q(transpose).
 *   3)R_solve Rx=Q(transpose)b.
 *
 * C Implementaion: Dr. I. Wilf.
 */
#define ORDER 8
#include<math.h>
#include "qrsolv.h"

int qrsolv (m, size, b)
double m[ORDER][ORDER];
int size ;
double b[ORDER];
{
    int i,
        j;
    double tau,
        m1[ORDER],
        m2[ORDER];

    if (qrdecomp (m, size, m1, m2) < 0) return 0 ;
        //error ("singularity in qrdecomp()\n");
    for (j = 0; j < (size - 1); j++) {
        tau = 0;
        for (i = j; i < size; i++)
            tau += m[i][j] * b[i];
        tau /= m1[j];
        for (i = j; i < size; i++)
            b[i] -= tau * m[i][j];
    }
    b[size - 1] /= m2[size - 1];
}

```

```

    rsolv (m, size, m2, b);
}
/*
*   Compute the QR decomposition of a square matrix m using the
Stewart
*   algorithm.
*   Upon termination, the decomposition is stored in m, m1 and m2 as
*   follows:
*   R is contained in the upper triangle of m except that its main
*   diagonal is contained in m2, and  $Q(transpos)=Q(n-1)*...*Q(1)$ 
*   where  $Q(j)=I-(U_j*U_j(transpos)/P_j)$  where
*    $U_j[i]=0, i=1 \rightarrow j-1, U_j[i]=m[i][j], i=j \rightarrow n, P_j=m1[j]$ .
*
*   Stewart, G.W., Introduction to matrix Computations, Academic Press,
*   New York (1973).
*
* C Implementaion: Dr. I. Wilf.
*/

```

```

int qrdecomp (m, size, m1, m2)
double m[ORDER][ORDER];
double m1[ORDER],
      m2[ORDER];
{
    int i,
        k,
        j;
    double eta,
           t,
           sigma,
           tau;

    for (k = 0; k < (size - 1); k++) {
        eta = 0.0;
        for (i = k; i < size; i++)
            if (fabs (m[i][k]) > eta)
                eta = fabs (m[i][k]);
        if (eta == 0.0)

```

```

        return (-1);
/* form Qk and premultiply m by it */
    t = 0;
    for (i = k; i < size; i++) {
        m[i][k] /= eta;
        t += m[i][k] * m[i][k];
    }
    if(m[k][k] >= 0.0)
        sigma = sqrt(t);
    else
        sigma = -sqrt(t);
    m[k][k] += sigma;
    m1[k] = sigma * m[k][k];
    m2[k] = (-eta * sigma);
    tau = 0;
    for (j = k + 1; j < size; j++) {
        tau = 0;
        for (i = k; i < size; i++)
            tau += m[i][k] * m[i][j];
        tau /= m1[k];
        for (i = k; i < size; i++)
            m[i][j] -= tau * m[i][k];
    }
}
m2[size - 1] = m[size - 1][size - 1];
return (0);
}
/*
 *   rsolv(m,size,m2,b)
 *   solve Rx=b for b, where the upper triangular matrix R is
 *   stored in M , m2.
 *
 * C Implementaion: Dr. I. Wilf.
 */

rsolv (m, size, m2, b)
double m[ORDER][ORDER];
double m2[ORDER],

```

```
        b[ORDER];  
    {  
        int    i,  
              j;  
        double s;  
        for (i = size - 2; i >= 0; i--) {  
            s = 0;  
            for (j = i + 1; j < size; j++)  
                s += m[i][j] * b[j];  
            b[i] = (b[i] - s) / m2[i];  
        }  
    }
```

```
#include <windows.h>
#include <windowsx.h>
#include <commdlg.h>
#include <stdlib.h>
#include <bios.h>
#include "const.h"
#include "bitmap.h"
#include "persp.h"
#include "lines.h"
#include "track.h"
#include "min_mag.h"
#include "lib.h"
```

```
int PASCAL I_cp_int_arr_to_RPOINT(RPOINT *Rpoint, int *ArrX, int *ArrY, int
num)
```

```
{
int i ;

for ( i = 0 ; i < num ; i++ ) {
    Rpoint[i].x = ArrX[i] ;
    Rpoint[i].y = ArrY[i] ;
}
```

```
return 1 ;
}
```

```
int PASCAL I_quad_in_new_origin(RPOINT *NewQuad, RPOINT *Quad,
int Xorg, int Yorg, int num)
```

```
{
int i ;

for ( i = 0 ; i < num ; i++ ) {
    NewQuad[i].x = Quad[i].x - Xorg ;
    NewQuad[i].y = Quad[i].y - Yorg ;
}
```

```
return 1 ;
}
```



```

int PASCAL l_copy_RPOINT_array(RPOINT *To,RPOINT *From,int num)
{
    int i ;

    for ( i = 0 ; i < num ; i++ ) {
        To[i] = From[i] ;
    }

    return 1;
}

int PASCAL l_copy_int_array(int *To,int *From,int num)
{
    int i ;

    for ( i = 0 ; i < num ; i++ ) {
        To[i] = From[i] ;
    }

    return 1;
}

int PASCAL l_find_bound_rect(RPOINT *Vertexes, RECT *Rectan)
{
    Rectan->left
    =(int)min(min(min(Vertexes[0].x,Vertexes[1].x),Vertexes[2].x),Vertexes[3].x) ;
    Rectan->top =
    (int)min(min(min(Vertexes[0].y,Vertexes[1].y),Vertexes[2].y),Vertexes[3].y) ;
    Rectan->right
    =(int)max(max(max(Vertexes[0].x,Vertexes[1].x),Vertexes[2].x),Vertexes[3].x) ;
    Rectan->bottom =
    (int)max(max(max(Vertexes[0].y,Vertexes[1].y),Vertexes[2].y),Vertexes[3].y) ;

    return 1 ;
}

```

```
int PASCAL find_extremes_in_1dim(RPOINT *Points, int Number, int Dim,  
                                double *MaxVal, double *MinVal)  
{  
    int i;  
    double MaxNum, MinNum;  
  
    if ( Dim == X_DIM ) {  
        MaxNum = MinNum = Points[0].x;  
        for ( i = 1 ; i < Number ; i++ ) {  
            if ( Points[i].x > MaxNum ) {  
                MaxNum = Points[i].x;  
                continue ;  
            }  
            if ( Points[i].x < MinNum ) {  
                MinNum = Points[i].x;  
                continue ;  
            }  
        }  
        *MaxVal = MaxNum ;  
        *MinVal = MinNum ;  
    }  
    if ( Dim == Y_DIM ) {  
        MaxNum = MinNum = Points[0].y ;  
        for ( i = 1 ; i < Number ; i++ ) {  
            if ( Points[i].y > MaxNum ) {  
                MaxNum = Points[i].y ;  
                continue ;  
            }  
            if ( Points[i].y < MinNum ) {  
                MinNum = Points[i].y ;  
                continue ;  
            }  
        }  
        *MaxVal = MaxNum ;  
        *MinVal = MinNum ;  
    }  
}
```

```

    return 1 ;
}

int PASCAL find_best_cluster(HWND hwnd, int Number, RPOINT *Points,
                             RPOINT *Cluster, int *NumCluster, int Dim,
                             int *Indexes, int *ClustIndex, RPOINT *NotCluster, int *NotIndexes)
{
    int i,j,k ;
    int Count,NCount ;
    int Distance ;
    int Range ;
    double MaxNum,MinNum ;
    int *DistArray ;
    double *SumArray ;
    double MaxSum ;
    double MaxNumbers,Value ;
    int MaxIndex ;

    if ( Number == 0 ) return 0 ;
    find_extremes_in_1dim(Points,Number,Dim,&MaxNum,&MinNum) ;
    Range = (int)(fabs(MaxNum - MinNum+0.5)) ;
    DistArray = (int *) GlobalAllocPtr (GMEM_MOVEABLE,
                                         (DWORD)sizeof(int)*(DWORD)Number*(DWORD)Range
    );
    SumArray = (double *) GlobalAllocPtr (GMEM_MOVEABLE,
                                         (DWORD)sizeof(double)*(DWORD)Number) ;

    for ( i = 0 ; i < Number ; i++ ) {
        SumArray[i] = 0.0 ;
        for ( j = 0 ; j < Range ; j++ ) {
            VALUE_IN(DistArray,i,j,Range) = 0 ;
        }
    }

    for ( i = 0 ; i < Number ; i++ ) {
        for ( j = 0 ; j < Number ; j++ ) {
            if ( i == j ) continue ;
            for ( k = 1 ; k <= Range ; k++ ) {
                if ( Dim == X_DIM ) {

```

```

        if ( fabs(Points[j].x - Points[i].x) <= (double)k ) {
            VALUE_IN(DistArray,i,k-1,Range) += 1 ;
        }
    } else {
        if ( fabs(Points[j].y - Points[i].y) <= (double)k ) {
            VALUE_IN(DistArray,i,k-1,Range) += 1 ;
        }
    }
}
}
}

MaxSum = 0.0 ;
MaxIndex = 0 ;
for ( i = 0 ; i < Number ; i++ ) {
    for ( j = 0 ; j < Range ; j++ ) {
        SumArray[i] += (double)(VALUE_IN(DistArray,i,j,Range))/(double)(j+1);
    }
    if ( SumArray[i] > MaxSum ) {
        MaxIndex = i ;
        MaxSum = SumArray[i] ;
    }
}

MaxNumbers = 0.0 ;

for ( i = 0 ; i < Range ; i++ ) {
    Value = (double)(VALUE_IN(DistArray,MaxIndex,i,Range))/(double)(i+1) ;
    if ( Value > MaxNumbers ) {
        MaxNumbers = Value ;
        Distance = i+1 ;
    }
}

if ( Range == 1 && MaxNumbers <= 2 ) {
    GlobalFreePtr(SumArray) ;
    GlobalFreePtr(DistArray) ;
    return 0 ;
}

```

```

}
if ( Range > 1 && MaxNumbers < (double)Number/((double)Range+0.01) ) {
    GlobalFreePtr(SumArray);
    GlobalFreePtr(DistArray);
    return 0;
}
Count = NCount = 0;
for ( j = 0; j < Number; j++ ) {
    if ( Dim == X_DIM ) {
        if ( fabs(Points[j].x - Points[MaxIndex].x) <= (double)Distance ) {
            Cluster[Count] = Points[j];
            ClustIndex[Count++] = Indexes[j];
        } else {
            NotCluster[NCount] = Points[j];
            NotIndexes[NCount++] = Indexes[j];
        }
    } else {
        if ( fabs(Points[j].y - Points[MaxIndex].y) <= (double)Distance ) {
            Cluster[Count] = Points[j];
            ClustIndex[Count++] = Indexes[j];
        } else {
            NotCluster[NCount] = Points[j];
            NotIndexes[NCount++] = Indexes[j];
        }
    }
}
}

```

```

*NumCluster = Count;
GlobalFreePtr(SumArray);
GlobalFreePtr(DistArray);

```

```

return 1;
}

```

```

int PASCAL print_transform(HFILE hFile, char * Head, Perspective_Transform
*Tp)
{

```

```

int j ;
char String[50] ;

    _lwrite(hFile,Head,strlen(Head)) ;
    for ( j = 0 ; j < 3 ; j++ ) {
        sprintf(String,"\\n%lf, %lf, %lf\\n",Tp->Pf[j][0],
                Tp->Pf[j][1],Tp->Pf[j][2]) ;
        _lwrite(hFile,String,strlen(String)) ;
    }
    return 1 ;
}

```

```

int PASCAL insert_new_vertexes(RPOINT *CurrVert, RPOINT *Vert1,
RPOINT *Vert2,
    RPOINT *Vert3, RPOINT *Vert4,double *Wheight,int num,RPOINT
*NewVert)
{
    int i ;
    RPOINT Delta[4] ;
    double Sum ;

    Sum =      Wheight[0] + Wheight[1] + Wheight[2] ;
    for ( i = 0 ; i < num-1 ; i++ ) {
        Vert1[i] = Vert1[i+1] ;
        Vert2[i] = Vert2[i+1] ;
        Vert3[i] = Vert3[i+1] ;
        Vert4[i] = Vert4[i+1] ;
    }
    Vert1[num-1] = CurrVert[0] ;
    Vert2[num-1] = CurrVert[1] ;
    Vert3[num-1] = CurrVert[2] ;
    Vert4[num-1] = CurrVert[3] ;

    for ( i = 0 ; i < num-1 ; i++ ) {
        Delta[i].x = Vert1[i+1].x - Vert1[i].x ;
        Delta[i].y = Vert1[i+1].y - Vert1[i].y ;
    }
}

```

```

Delta[2].x =(Delta[2].x*Wheight[2] + Delta[1].x*Wheight[1] +
              Delta[0].x*Wheight[0])/Sum ;
Delta[2].y =(Delta[2].y*Wheight[2] + Delta[1].y*Wheight[1] +
              Delta[0].y*Wheight[0])/Sum ;
NewVert[0].x = Vert1[2].x + Delta[2].x ;
NewVert[0].y = Vert1[2].y + Delta[2].y ;

```

```

for ( i = 0 ; i < num-1 ; i++ ) {
    Delta[i].x = Vert2[i+1].x - Vert2[i].x ;
    Delta[i].y = Vert2[i+1].y - Vert2[i].y ;
}
Delta[2].x =(Delta[2].x*Wheight[2] + Delta[1].x*Wheight[1] +
              Delta[0].x*Wheight[0])/Sum ;
Delta[2].y =(Delta[2].y*Wheight[2] + Delta[1].y*Wheight[1] +
              Delta[0].y*Wheight[0])/Sum ;
NewVert[1].x = Vert2[2].x + Delta[2].x ;
NewVert[1].y = Vert2[2].y + Delta[2].y ;

```

```

for ( i = 0 ; i < num-1 ; i++ ) {
    Delta[i].x = Vert3[i+1].x - Vert3[i].x ;
    Delta[i].y = Vert3[i+1].y - Vert3[i].y ;
}
Delta[2].x =(Delta[2].x*Wheight[2] + Delta[1].x*Wheight[1] +
              Delta[0].x*Wheight[0])/Sum ;
Delta[2].y =(Delta[2].y*Wheight[2] + Delta[1].y*Wheight[1] +
              Delta[0].y*Wheight[0])/Sum ;
NewVert[2].x = Vert3[2].x + Delta[2].x ;
NewVert[2].y = Vert3[2].y + Delta[2].y ;

```

```

for ( i = 0 ; i < num-1 ; i++ ) {
    Delta[i].x = Vert4[i+1].x - Vert4[i].x ;
    Delta[i].y = Vert4[i+1].y - Vert4[i].y ;
}
Delta[2].x =(Delta[2].x*Wheight[2] + Delta[1].x*Wheight[1] +
              Delta[0].x*Wheight[0])/Sum ;
Delta[2].y =(Delta[2].y*Wheight[2] + Delta[1].y*Wheight[1] +
              Delta[0].y*Wheight[0])/Sum ;
NewVert[3].x = Vert4[2].x + Delta[2].x ;

```

```
NewVert[3].y = Vert4[2].y + Delta[2].y ;
```

```
return 1 ;
```

```
}
```

```
int PASCAL transform_rpoint_arr(RPOINT *SrcPnts, RPOINT *DstPnts,  
                                int num ,Perspective_Transform TpCurr )
```

```
{
```

```
double DstX,DstY,w ;
```

```
int i ;
```

```
for ( i = 0 ; i < num ; i++ ) {
```

```
    DstX = SrcPnts[i].x * TpCurr.Pf[0][0] +
```

```
           SrcPnts[i].y * TpCurr.Pf[1][0] + TpCurr.Pf[2][0] ;
```

```
    DstY = SrcPnts[i].x * TpCurr.Pf[0][1] +
```

```
           SrcPnts[i].y * TpCurr.Pf[1][1] + TpCurr.Pf[2][1] ;
```

```
    w = SrcPnts[i].x * TpCurr.Pf[0][2] +
```

```
         SrcPnts[i].y * TpCurr.Pf[1][2] + TpCurr.Pf[2][2] ;
```

```
    DstPnts[i].x = DstX/w ;
```

```
    DstPnts[i].y = DstY/w ;
```

```
}
```

```
return 1 ;
```

```
}
```



```

#include <windows.h>
#include <windowsx.h>
#include <commdlg.h>
#include <stdlib.h>
#include "const.h"
#include "bitmap.h"
#include "lines.h"
#include "track.h"
#include "persp.h"
#include "min_mag.h"
#include "lib.h"

```

```

int PASCAL improve_diff_bmap(MYBITMAP *,MYBITMAP *);
int PASCAL copy_rect_bmap(int ,int ,MYBITMAP *,MYBITMAP *);

```

```

int PASCAL perspective(HWND hwnd,MYBITMAP FAR *SrcBmap,
MYBITMAP FAR *DstBmap,
    RPOINT* SrcPnts,RPOINT * DstPnts,int ColorModel,
    Perspective_Transform *Tp)
{
    //Rectan2Quad(hwnd,SrcPnts, DstPnts,Tp);
    Quad2Quad(hwnd,SrcPnts, DstPnts,Tp);

    Perspective_map(SrcBmap, Tp, DstBmap, DstPnts,ColorModel);
    return TRUE ;
}

```

```

int PASCAL perspective_near(HWND hwnd,MYBITMAP FAR *SrcBmap,
MYBITMAP FAR *DstBmap,
    RPOINT* SrcPnts,RPOINT * DstPnts,int ColorModel,
    Perspective_Transform *Tp)
{
    //Rectan2Quad(hwnd,SrcPnts, DstPnts,Tp);
    Quad2Quad(hwnd,SrcPnts, DstPnts,Tp);

    Perspective_near_map(SrcBmap, Tp, DstBmap, DstPnts,ColorModel);

```

```

    return TRUE ;
}

int PASCAL Rectan2Quad(HWND hwnd, RPOINT *src_pts, RPOINT
*dst_pts,
                        Perspective_Transform *Tp)
{
    double          x0, y0, x1, y1, x2, y2, x3, y3;
    double          dx1, dy1, dx2, dy2, dx3, dy3;
    double          denom ;

    double          a11, a12, a13, a21, a22, a23, a31, a32, a33;

    double          A[3][3], B[3][3];

    /* Verify that src_pts do form a rectangle */
    if ( check_if_rect(src_pts) == FALSE ) {
        MessageBox (hwnd, "Source is not a rectangle",
            "Perspective trans.", MB_ICONEXCLAMATION | MB_OK);
        return NULL ;
    }
    /* Solve for transformation from [(0,0),(1,1)] -> Quad */
    x0 = dst_pts[0].x;
    y0 = dst_pts[0].y;
    x1 = dst_pts[1].x;
    y1 = dst_pts[1].y;
    x2 = dst_pts[2].x;
    y2 = dst_pts[2].y;
    x3 = dst_pts[3].x;
    y3 = dst_pts[3].y;

    dx1 = x1 - x2;
    dy1 = y1 - y2;
    dx2 = x3 - x2;
    dy2 = y3 - y2;
    dx3 = x0 - x1 + x2 - x3;
    dy3 = y0 - y1 + y2 - y3;

```

```

/* if dx3 = dy3 = 0: transformation is affine */

/* otherwise:      transformation is perspective */

/*
 * Forward transformation:
 *
 * [x_,y_,w_] = [u,v,1] Pf
 *
 * where: [a11 a12 a13] Pf = [a21 a22 a23] [a31 a32 a33]
 *
 * Then: [x, y] = [x_/w_, y_/w_]
 */

```

```

denom = det2(dx1, dx2, dy1, dy2);

```

```

a13 = det2(dx3, dx2, dy3, dy2) / denom;
a23 = det2(dx1, dx3, dy1, dy3) / denom;
a11 = x1 - x0 + a13 * x1;
a21 = x3 - x0 + a23 * x3;
a31 = x0;
a12 = y1 - y0 + a13 * y1;
a22 = y3 - y0 + a23 * y3;
a32 = y0;
a33 = 1.0;

```

```

{
  A[0][0] = a11;
  A[0][1] = a12;
  A[0][2] = a13;
  A[1][0] = a21;
  A[1][1] = a22;
  A[1][2] = a23;
  A[2][0] = a31;
  A[2][1] = a32;
  A[2][2] = a33;
}

```

```

/* pre-multiply by the matrix converting src_pts->[(0,0),(1,1)] */
{
    double    xs = src_pts[0].x;
    double    ys = src_pts[0].y;
    double    xe = src_pts[2].x;
    double    ye = src_pts[2].y;
    double    dx = xe - xs;
    double    dy = ye - ys;
    B[0][0] = 1.0 / dx;
    B[1][0] = 0.0;
    B[2][0] = -xs / dx;
    B[0][1] = 0.0;
    B[1][1] = 1.0 / dy;
    B[2][1] = -ys / dy;
    B[0][2] = 0.0;
    B[1][2] = 0.0;
    B[2][2] = 1.0;

    multmat(hwnd,3, 3, 3, (double*)B, (double*)A, (double*)(Tp->Pf));
}

/*
 * Backward transformation:
 *
 * [u_,v_,w_] = [x,y,1] Pb
 *
 * where: [A11 A12 A13] Pb = [A21 A22 A23] [A31 A32 A33]
 *
 * is the inverse of Pf.
 *
 * Then: [u, v] = [u_/w_, v_/w_]
 */
{
    a11 = Tp->Pf[0][0];
    a12 = Tp->Pf[0][1];
    a13 = Tp->Pf[0][2];

```

```

        a21 = Tp->Pf[1][0];
        a22 = Tp->Pf[1][1];
        a23 = Tp->Pf[1][2];
        a31 = Tp->Pf[2][0];
        a32 = Tp->Pf[2][1];
        a33 = Tp->Pf[2][2];
    }

    Tp->Pb[0][0] = a22 * a33 - a23 * a32;
    Tp->Pb[0][1] = a13 * a32 - a12 * a33;
    Tp->Pb[0][2] = a12 * a23 - a13 * a22;
    Tp->Pb[1][0] = a23 * a31 - a21 * a33;
    Tp->Pb[1][1] = a11 * a33 - a13 * a31;
    Tp->Pb[1][2] = a13 * a21 - a11 * a23;
    Tp->Pb[2][0] = a21 * a32 - a22 * a31;
    Tp->Pb[2][1] = a12 * a31 - a11 * a32;
    Tp->Pb[2][2] = a11 * a22 - a12 * a21;

    return 1 ;
}

int PASCAL Quad2Rectan(HWND hwnd, RPOINT *src_pts, RPOINT
*dst_pts,
                        Perspective_Transform *Tp)
{
    double Tmp ;

    Rectan2Quad(hwnd,dst_pts,src_pts,Tp) ;

    Tmp = Tp->Pf[0][0] ;
    Tp->Pf[0][0] = Tp->Pb[0][0] ;
    Tp->Pb[0][0] = Tmp ;

    Tmp = Tp->Pf[0][1] ;
    Tp->Pf[0][1] = Tp->Pb[0][1] ;
    Tp->Pb[0][1] = Tmp ;

    Tmp = Tp->Pf[0][2] ;

```

```

Tp->Pf[0][2] = Tp->Pb[0][2] ;
Tp->Pb[0][2] = Tmp ;

```

```

Tmp = Tp->Pf[1][0] ;
Tp->Pf[1][0] = Tp->Pb[1][0] ;
Tp->Pb[1][0] = Tmp ;

```

```

Tmp = Tp->Pf[1][1] ;
Tp->Pf[1][1] = Tp->Pb[1][1] ;
Tp->Pb[1][1] = Tmp ;

```

```

Tmp = Tp->Pf[1][2] ;
Tp->Pf[1][2] = Tp->Pb[1][2] ;
Tp->Pb[1][2] = Tmp ;

```

```

Tmp = Tp->Pf[2][0] ;
Tp->Pf[2][0] = Tp->Pb[2][0] ;
Tp->Pb[2][0] = Tmp ;

```

```

Tmp = Tp->Pf[2][1] ;
Tp->Pf[2][1] = Tp->Pb[2][1] ;
Tp->Pb[2][1] = Tmp ;

```

```

Tmp = Tp->Pf[2][2] ;
Tp->Pf[2][2] = Tp->Pb[2][2] ;
Tp->Pb[2][2] = Tmp ;

```

```

return 1;
}

```

```

int PASCAL check_if_rect(RPOINT* Points)
{
    if ( Points[0].x != Points[3].x ) return FALSE ;
    if ( Points[1].x != Points[2].x ) return FALSE ;
    if ( Points[0].y != Points[1].y ) return FALSE ;
    if ( Points[2].y != Points[3].y ) return FALSE ;

```

```

    return TRUE ;
}

int PASCAL Perspective_near_map(MYBITMAP
*src_bmap,Perspective_Transform* Tp,
    MYBITMAP *dst_bmap, RPOINT *dst_pts,int ColorModel)
{
    RECT      r,Screen;
    DWORD      i, j;
    DWORD      Size ;
    BYTE  huge *Ptr ;
    double      RcBil ;
    int Colsmin1,Rowsmin1 ;
    double      y_dst ;
    double      x_dst ;
    RPOINT      uv, xy;
    double      D ;
    BYTE  huge *Colors ;
    int      uy,ux ;

    /* Find bounding rectangle of dst_pts */
    l_find_bound_rect(dst_pts, &r) ;

    Colsmin1 = src_bmap->cols -1 ;
    Rowsmin1 = src_bmap->rows -1 ;
    Colors = (BYTE huge*)GlobalAllocPtr(GMEM_MOVEABLE,3) ;

    for (i = r.top; i < r.bottom; i++) {
        y_dst = (double) i;
        for (j = r.left; j < r.right; j++) {
            x_dst = (double) j;
            D = 1.0;

            xy.x = x_dst;
            xy.y = y_dst;

            uv = bPerspective(xy, Tp);
            if (uv.x < 0.0 || uv.x >= (double) (src_bmap->cols-1))

```

345

SUBSTITUTE SHEET (RULE 26)

```

        continue;
    if (uv.y < 0.0 || uv.y >= (double) (src_bmap->rows-1))
        continue;

    uy = (int)(uv.y+0.5);
    ux = (int)(uv.x+0.5);
    if ( ColorModel == GREY_MODEL ) {
        RcBil = BITMAP_PLACE(src_bmap,uy,ux);
        BITMAP_PLACE(dst_bmap,i,j) = (BYTE)RcBil;
    } else { // COLOR_MODEL
        Ptr = BITMAP_RGB_PLACE_PTR(dst_bmap,i,j);
        Colors = BITMAP_RGB_PLACE_PTR(src_bmap,uy,ux);
        Ptr[0] = (BYTE)Colors[0];
        Ptr[1] = (BYTE)Colors[1];
        Ptr[2] = (BYTE)Colors[2];
    }
}
}
GlobalFreePtr(Colors);
return (0);
}

```

```

int PASCAL Perspective_map(MYBITMAP FAR
*src_bmap,Perspective_Transform* Tp,
MYBITMAP FAR *dst_bmap, RPOINT *dst_pts,int ColorModel)
{
    RECT      r,Screen;
    DWORD      i,j;
    DWORD      Size;
    BYTE  huge *Ptr;
    double      RcBil;
    int Colsmin1,Rowsmin1;
    double      y_dst;
    double      x_dst;
    RPOINT      uv, xy;
    double      D;
    double      Colors[3];

```



```

/* Find bounding rectangle of dst_pts */
l_find_bound_rect(dst_pts, &r);

Colsmin1 = src_bmap->cols -1 ;
Rowsmin1 = src_bmap->rows -1 ;

for (i = r.top; i < r.bottom; i++) {
    y_dst = (double) i;
    for (j = r.left; j < r.right; j++) {
        x_dst = (double) j;
        D = 1.0;

        xy.x = x_dst;
        xy.y = y_dst;

        uv = bPerspective(xy, Tp);
        if (uv.x < 0.0 || uv.x >= (double) (src_bmap->cols))
            continue;
        if (uv.y < 0.0 || uv.y >= (double) (src_bmap->rows))
            continue;

        if ( ColorModel == GREY_MODEL ) {
            RcBil = (BYTE)bilinear(uv.x,uv.y, src_bmap);
            if ( RcBil == -1.0 ) continue ;
            BITMAP_PLACE(dst_bmap,i,j) = (BYTE)RcBil ;
        } else {      // COLOR_MODEL
            Ptr = BITMAP_RGB_PLACE_PTR(dst_bmap,i,j) ;
            bilinear_rgb(uv.x,uv.y,src_bmap,Colors) ;
            Ptr[0] = (BYTE)Colors[0] ;
            Ptr[1] = (BYTE)Colors[1] ;
            Ptr[2] = (BYTE)Colors[2] ;
        }
    }
}
return (0);
}

```

```

RPOINT bPerspective(RPOINT xy, Perspective_Transform *Tp)
{
    RPOINT    uv;
    double    x = xy.x;
    double    y = xy.y;
    double    u_ = x * Tp->Pb[0][0] + y * Tp->Pb[1][0] + Tp->Pb[2][0];
    double    v_ = x * Tp->Pb[0][1] + y * Tp->Pb[1][1] + Tp->Pb[2][1];
    double    w_ = x * Tp->Pb[0][2] + y * Tp->Pb[1][2] + Tp->Pb[2][2];

    uv.x = u_ / w_;
    uv.y = v_ / w_;
    return (uv);
}

```

```

RPOINT PASCAL fPerspective(RPOINT xy, Perspective_Transform *Tp)
{
    RPOINT    uv;
    double    x = xy.x;
    double    y = xy.y;
    double    u_ = x * Tp->Pf[0][0] + y * Tp->Pf[1][0] + Tp->Pf[2][0];
    double    v_ = x * Tp->Pf[0][1] + y * Tp->Pf[1][1] + Tp->Pf[2][1];
    double    w_ = x * Tp->Pf[0][2] + y * Tp->Pf[1][2] + Tp->Pf[2][2];

    uv.x = u_ / w_;
    uv.y = v_ / w_;
    return (uv);
}

```

```

/*
double PASCAL dPerspective(RPOINT xy, Perspective_Transform *Tp)
{
    double    Du, Dv, D;
    double    Dux, Duy;
    double    Dvx, Dvy;

    double    x = xy.x;
    double    y = xy.y;

```

```

double      u_ = x * Tp->Pb[0][0] + y * Tp->Pb[1][0] + Tp->Pb[2][0];
double      v_ = x * Tp->Pb[0][1] + y * Tp->Pb[1][1] + Tp->Pb[2][1];
double      w_ = x * Tp->Pb[0][2] + y * Tp->Pb[1][2] + Tp->Pb[2][2];

double      u_x = Tp->Pb[0][0];
double      u_y = Tp->Pb[1][0];
double      v_x = Tp->Pb[0][1];
double      v_y = Tp->Pb[1][1];
double      w_x = Tp->Pb[0][2];
double      w_y = Tp->Pb[1][2];

Dux = (u_x * w_ - u_ * w_x) / (w_ * w_);
Duy = (u_y * w_ - u_ * w_y) / (w_ * w_);
Du = sqrt(Dux * Dux + Duy * Duy);

Dvx = (v_x * w_ - v_ * w_x) / (w_ * w_);
Dvy = (v_y * w_ - v_ * w_y) / (w_ * w_);
Dv = sqrt(Dvx * Dvx + Dvy * Dvy);

D = max(Du, Dv);

return (D);
}
*/

```

```
#define MEDIAN_EDGE 5
```

```
#define MEDIAN_SIDE 2
```

```

int PASCAL median_filter_5(HWND hwnd,MYBITMAP *Bmap)
{
int i, j, k, l, n;
int MedVec[MEDIAN_EDGE*MEDIAN_EDGE];
BYTE IndexArr[256];
int Median;
MYBITMAP *TmpBmap;
BYTE huge *TmpPtr;

```

```

BYTE huge *BmapPtr ;
DWORD Size,PI ;
DWORD FromTime,ToTime ;
char String[100] ;
int Sum ;
int RowLimit , ColLimit ;

TmpBmap = bm_alloc(Bmap->cols,Bmap->rows,GREY_MODEL) ;
Size = (DWORD)Bmap->cols*(DWORD)Bmap->rows ;
TmpBmap->gpic = (BYTE huge*)GlobalAllocPtr(GMEM_MOVEABLE,Size) ;
BmapPtr = Bmap->gpic ;
TmpPtr = TmpBmap->gpic ;
for ( PI = 0 ; PI < Size ; PI++ ) {
    *(TmpPtr++) = *(BmapPtr++) ;
}
for ( i = 0 ; i < Bmap->rows ; i++ ) {
    for ( j = 0 ; j < Bmap->cols ; j++ ) {
        if ( BITMAP_PLACE(TmpBmap,i,j) <= 24 ) {
            BITMAP_PLACE(TmpBmap,i,j) = 0 ;
        }
    }
}
WriteGreyRgb (hwnd,"no48.rgb",TmpBmap) ;
//FromTime = GetTickCount() ;

RowLimit = Bmap->rows -3 ;
for ( i = 2 ; i < RowLimit ; i++ ) {
    for ( k = 0 ; k < 256 ; k++ ) IndexArr[k] =0 ;
    for ( j = 0 ; j < 5 ; j++ ) {
        IndexArr[BITMAP_PLACE(TmpBmap,i-2,j)] += 1 ;
        IndexArr[BITMAP_PLACE(TmpBmap,i-1,j)] += 1 ;
        IndexArr[BITMAP_PLACE(TmpBmap,i,j)] += 1 ;
        IndexArr[BITMAP_PLACE(TmpBmap,i+1,j)] += 1 ;
        IndexArr[BITMAP_PLACE(TmpBmap,i+2,j)] += 1 ;
    }
    for ( Sum = k = 0 ; k < 256 ; k++ ) {
        Sum += IndexArr[k] ;
        if ( Sum >= 13 ) {

```

```

        BITMAP_PLACE(TmpBmap,i,2) = k ;
        break ;
    }
}
ColLimit = Bmap->cols - 3;
for ( j = 3 ; j < ColLimit ; j++ ) {
    IndexArr[BITMAP_PLACE(TmpBmap,i-2,j-3)] -= 1 ;
    IndexArr[BITMAP_PLACE(TmpBmap,i-1,j-3)] -= 1 ;
    IndexArr[BITMAP_PLACE(TmpBmap,i,j-3)] -= 1 ;
    IndexArr[BITMAP_PLACE(TmpBmap,i+1,j-3)] -= 1 ;
    IndexArr[BITMAP_PLACE(TmpBmap,i+2,j-3)] -= 1 ;

    IndexArr[BITMAP_PLACE(TmpBmap,i-2,j+2)] += 1 ;
    IndexArr[BITMAP_PLACE(TmpBmap,i-1,j+2)] += 1 ;
    IndexArr[BITMAP_PLACE(TmpBmap,i,j+2)] += 1 ;
    IndexArr[BITMAP_PLACE(TmpBmap,i+1,j+2)] += 1 ;
    IndexArr[BITMAP_PLACE(TmpBmap,i+2,j+2)] += 1 ;
    for ( Sum = k = 0 ; k < 256 ; k++ ) {
        Sum += IndexArr[k] ;
        if ( Sum >= 13 ) {
            BITMAP_PLACE(TmpBmap,i,j) = k ;
            break ;
        }
    }
}
}
WriteGreyRgb (hwnd,"med5.rgb",TmpBmap) ;

//ToTime = GetTickCount() ;
//sprintf(String,"Time elapsed = %lu",ToTime-FromTime) ;
//MessageBox (hwnd, String,"Median Calc", MB_ICONEXCLAMATION |
MB_OK) ;
improve_diff_bmap(Bmap,TmpBmap) ;
bm_free(TmpBmap) ;
return 1;
}

int PASCAL improve_diff_bmap(MYBITMAP *Bmap,MYBITMAP *TmpBmap)
351

```

```

{
    int i,j;
    int ICount,JCount;
    int Frj;
    int Limit;
    int Cols,Rows;

    Frj = 0;
    JCount = 0;
    Cols = Bmap->cols;
    Rows = Bmap->rows;
    Limit = Bmap->rows / 8;
    for ( j = 0; j < Cols; j++ ) {
        ICount = 0;
        for ( i = 0; i < Rows; i++ ) {
            if ( BITMAP_PLACE(TmpBmap,i,j) != 0 ) ICount++;
        }
        if ( ICount < Limit ) JCount++;
        else {
            if ( JCount > 0 ) {
                copy_rect_bmap(Frj,JCount,TmpBmap,Bmap);
                JCount = 0;
            }
            Frj = j+1;
        }
    }

    return 1;
}

int PASCAL copy_rect_bmap(int FromCol,int NumCols,MYBITMAP
*From,MYBITMAP *To)
{
    int i,j;
    int Until;

    Until = FromCol + NumCols;
    for ( j = FromCol; j < Until; j++ ) {

```

```

    for ( i = 0 ; i < From->rows ; i++ ) {
        BITMAP_PLACE(To,i,j) = BITMAP_PLACE(From,i,j) ;
    }
}

return 1 ;
}

```

```

int PASCAL get_tresh_for_occ(MYBITMAP *Bmap,int *Tresh)
{
    DWORD Size,n ;
    BYTE huge *TmpPic ;
    DWORD IndexArr[256] ;
    int i ;
    DWORD Sum, MidSize ;

    for ( i = 0 ; i < 256 ; i++ ) {
        IndexArr[i] = 0 ;
    }
    Size = (DWORD)(Bmap->cols)*(DWORD)(Bmap->rows) ;
    TmpPic = Bmap->gpPic ;
    for ( n = 1 ; n < Size ; n++ ) {
        IndexArr[*(TmpPic++)] += 1 ;
    }
    MidSize = Size/2 ;
    Sum = 0 ;
    for ( i = 0 ; i < 256 ; i++ ) {
        Sum += IndexArr[i] ;
        if ( Sum >= MidSize ) {
            *Tresh = i ;
            break ;
        }
    }
    if ( *Tresh < 32 ) *Tresh = 32 ;
    return 1 ;
}

```

```

int PASCAL perspective_mask(HWND hwnd,MYBITMAP
*SrcBmap,MYBITMAP *DstBmap,
    MYBITMAP * Alpha,RPOINT *SrcPnts, RPOINT *DstPnts, int
ColorModel,
    int interpolation,MYBITMAP *MaskBmap)
{
    Perspective_Transform Tp ;

    //Rectan2Quad(hwnd,SrcPnts, DstPnts,&Tp);
    Quad2Quad(hwnd,SrcPnts, DstPnts,&Tp);

    Perspective_map_mask(SrcBmap, &Tp, DstBmap, Alpha,
DstPnts,ColorModel,
    interpolation,MaskBmap);

    return 1 ;
}

int PASCAL Perspective_map_mask(MYBITMAP FAR
*src_bmap,Perspective_Transform* Tp,
    MYBITMAP FAR *dst_bmap, MYBITMAP * Alpha,RPOINT *dst_pts,int
ColorModel,
    int interpolation,MYBITMAP *MaskBmap)
{
    RECT    r,Screen;
    DWORD    i, j;
    DWORD    Size ;
    BYTE    huge *Ptr ;
    double    RcBil ;
    double    Colors[3] ;
    int        AlphaNumber ;

    /* Find bounding rectangle of dst_pts */
    l_find_bound_rect(dst_pts, &r) ;

    for (i = r.top; i < r.bottom; i++) {
        double    y_dst = (double) i;

```



```

for (j = r.left; j < r.right; j++) {
    double      x_dst = (double) j;
    RPOINT      uv, xy;
    double      D = 1.0;

    xy.x = x_dst;
    xy.y = y_dst;

    uv = bPerspective(xy, Tp);
    if (uv.x < 0.0 || uv.x >= (double) (src_bmap->cols))
        continue;
    if (uv.y < 0.0 || uv.y >= (double) (src_bmap->rows))
        continue;

    if ( ColorModel == GREY_MODEL ) {
        RcBil = (BYTE)bilinear(uv.x,uv.y, src_bmap);
        if ( RcBil == -1.0 ||
            BITMAP_PLACE(MaskBmap,i-r.top,j-r.left) == 0 )
            continue ;
        BITMAP_PLACE(dst_bmap,i,j) = (BYTE)RcBil ;
    } else {      // COLOR_MODEL
        Ptr = BITMAP_RGB_PLACE_PTR(dst_bmap,i,j) ;
        RcBil = bilinear_rgb(uv.x,uv.y,src_bmap,Colors) ;
        if ( RcBil == -1.0 ||
            BITMAP_PLACE(MaskBmap,i-r.top,j-r.left) == 0 ) continue
    }

    if (BITMAP_PLACE(MaskBmap,i-r.top,j-r.left) == 2 ) {
        Ptr[0] =
        (BYTE)((double)((DWORD)Colors[0]*(DWORD)128 +
                        (DWORD)Ptr[0]*(DWORD)128)/256.0) ;
        Ptr[1] =
        (BYTE)((double)((DWORD)Colors[1]*(DWORD)128 +
                        (DWORD)Ptr[1]*(DWORD)128)/256.0) ;
        Ptr[2] =
        (BYTE)((double)((DWORD)Colors[2]*(DWORD)128 +
                        (DWORD)Ptr[2]*(DWORD)128)/256.0) ;
    }
}

```

```

        continue ;
    }

    AlphaNumber = BITMAP_PLACE(Alpha,i,j) ;
    if ( AlphaNumber > 1) {
        Ptr[0] =
            (BYTE)((double)((DWORD)Colors[0]*(DWORD)(256-
AlphaNumber) +
            (DWORD)Ptr[0]*(DWORD)AlphaNumber)/256.0) ;
        Ptr[1] =
            (BYTE)((double)((DWORD)Colors[1]*(DWORD)(256-
AlphaNumber) +
            (DWORD)Ptr[1]*(DWORD)AlphaNumber)/256.0) ;
        Ptr[2] =
            (BYTE)((double)((DWORD)Colors[2]*(DWORD)(256-
AlphaNumber) +
            (DWORD)Ptr[2]*(DWORD)AlphaNumber)/256.0) ;

        } else {
            Ptr[0] = (BYTE)Colors[0] ;
            Ptr[1] = (BYTE)Colors[1] ;
            Ptr[2] = (BYTE)Colors[2] ;
        }
    }
}
}
return (0);
}

```

```

int PASCAL perspective_al(HWND hwnd,MYBITMAP FAR *SrcBmap,
    MYBITMAP FAR *DstBmap,MYBITMAP *Alpha,
    RPOINT* SrcPnts,RPOINT * DstPnts,int ColorModel,
    Perspective_Transform *Tp)
{

```

```

    //Rectan2Quad(hwnd,SrcPnts, DstPnts,Tp);

```

356

```

Quad2Quad(hwnd,SrcPnts, DstPnts,Tp);

Perspective_map_al(SrcBmap, Tp, DstBmap, Alpha,DstPnts,ColorModel);
return TRUE ;
}

```

```

int PASCAL Perspective_map_al(MYBITMAP
*src_bmap,Perspective_Transform *Tp,
MYBITMAP *dst_bmap, MYBITMAP *Alpha,RPOINT *dst_pts,int ColorModel)
{
RECT      r,Screen;
DWORD      i, j;
DWORD      Size ;
BYTE  huge *Ptr ;
double      RcBil ;
int Colsm1,Rowsm1 ;
double      y_dst ;
double      x_dst ;
RPOINT      uv, xy;
double      D ;
double      Colors[3] ;
int      AlphaNumber ;

```

```

/* Find bounding rectangle of dst_pts */

```

```

l_find_bound_rect(dst_pts, &r) ;

```

```

Colsm1 = src_bmap->cols -1 ;

```

```

Rowsm1 = src_bmap->rows -1 ;

```

```

for (i = r.top; i < r.bottom; i++) {

```

```

    y_dst = (double) i;

```

```

    for (j = r.left; j < r.right; j++) {

```

```

        x_dst = (double) j;

```

```

        D = 1.0;

```

```

        xy.x = x_dst;

```

```

        xy.y = y_dst;

```

```

    uv = bPerspective(xy, Tp);
    if (uv.x < 0.0 || uv.x >= (double) (src_bmap->cols))
        continue;
    if (uv.y < 0.0 || uv.y >= (double) (src_bmap->rows))
        continue;
    if ( BITMAP_PLACE(Alpha,i,j) == 0 ) continue ;

    if ( ColorModel == GREY_MODEL ) {
        RcBil = (BYTE)bilinear(uv.x,uv.y, src_bmap);
        if ( RcBil == -1.0 ) continue ;
        BITMAP_PLACE(dst_bmap,i,j) = (BYTE)RcBil ;
    } else {      // COLOR_MODEL
        Ptr = BITMAP_RGB_PLACE_PTR(dst_bmap,i,j) ;
        bilinear_rgb(uv.x,uv.y,src_bmap,Colors) ;
        AlphaNumber = BITMAP_PLACE(Alpha,i,j) ;
        if ( AlphaNumber > 1) {
            Ptr[0] =
                (BYTE)((double)((DWORD)Colors[0]*(DWORD)(256-
AlphaNumber) +
                (DWORD)Ptr[0]*(DWORD)AlphaNumber)/256.0) ;
            Ptr[1] =
                (BYTE)((double)((DWORD)Colors[1]*(DWORD)(256-
AlphaNumber) +
                (DWORD)Ptr[1]*(DWORD)AlphaNumber)/256.0) ;
            Ptr[2] =
                (BYTE)((double)((DWORD)Colors[2]*(DWORD)(256-
AlphaNumber) +
                (DWORD)Ptr[2]*(DWORD)AlphaNumber)/256.0) ;

        } else {
            Ptr[0] = (BYTE)Colors[0] ;
            Ptr[1] = (BYTE)Colors[1] ;
            Ptr[2] = (BYTE)Colors[2] ;
        }
    }
}
}
return (0);

```

}

```
#include <windows.h>
#include <windowsx.h>
#include <commdlg.h>
#include <stdlib.h>
#include <bios.h>
#include "const.h"
#include "bitmap.h"
#include "persp.h"
#include "lines.h"
#include "track.h"
#include "min_mag.h"
#include "lib.h"
```

```
int PASCAL get_quad_segment_on_x(LINE ,LINE ,double ,double ,int
,RPOINT *) ;
```

```
int PASCAL mkline(double x1, double y1, double x2, double y2,LINE *l)
{
```

```
    l->a = y2 - y1;
    l->b = x1 - x2;
    l->c = -l->a * x1 -l->b * y1;
```

```
    return(1);
```

```
}
```

```
int PASCAL mksegment(double x1, double y1, double x2, double y2,SEG *s)
```

```
{
```

```
    s->a = y2 - y1;
    s->b = x1 - x2;
    s->c = -s->a * x1 -s->b * y1;
    s->xmax = max(x1,x2) ;
    s->xmin = min(x1,x2) ;
    s->ymax = max(y1,y2) ;
    s->ymin = min(y1,y2) ;
```

```
    return 1 ;
```

}

```
int PASCAL get_point_on_segment(SEG s1, double param, RPOINT *Point)
```

{

```
if ( param < 0.0 || param > 1.0 ) return 0 ;
```

```
Point->x = param * (s1.xmax-s1.xmin) + s1.xmin ;
```

```
Point->y = param * (s1.ymax-s1.ymin) + s1.ymin ;
```

```
return 1 ;
```

}

```
//this function returns in DstPnts the points calculated relative to the
//Xparam and YParam always from the minimum.
```

```
int PASCAL get_shrunked_yquad(RPOINT *SrcPnts, double param, RPOINT
*DstPnts)
```

{

```
SEG seg1,seg2,seg3,seg4 ;
```

```
int i ;
```

```
for ( i = 0 ; i < 4 ; i++ ) {
```

```
    DstPnts[i].x = SrcPnts[i].x ;
```

```
    DstPnts[i].y = SrcPnts[i].y ;
```

}

```
mksegment(SrcPnts[1].x, SrcPnts[1].y, SrcPnts[2].x, SrcPnts[2].y, &seg2) ;
```

```
mksegment(SrcPnts[3].x, SrcPnts[3].y, SrcPnts[0].x, SrcPnts[0].y, &seg4) ;
```

```
if ( get_point_on_segment(seg2, param, &(DstPnts[2])) == 0 ) return 0 ;
```

```
get_point_on_segment(seg4, param, &(DstPnts[3])) ;
```

```
return 1 ;
```

}

```
int PASCAL isect_lines(LINE l1, LINE l2, RPOINT *p)
```

{

```
double x,y,w,eps ;
```

```

x = l1.b * l2.c - l2.b * l1.c;
y = l1.c * l2.a - l2.c * l1.a;
w = l1.a * l2.b - l2.a * l1.b;
eps = 0.0001;
```

```

if(fabs(w) <= eps)
    return(0);
p->x = x / w;
p->y = y / w;
return(1);
}
```

```
int PASCAL center_of_bounded_rect(RPOINT p1,RPOINT p2, RPOINT p3,
RPOINT p4,
```

```
    RPOINT *Center,double *Xwidth,double *Ywidth)
```

```

{
RPOINT DownCorner,UpCorner ;
RPOINT LeftCorner,RightCorner ;
```

```

if ( p2.y > p1.y ) {
    UpCorner.x = p2.x;
    UpCorner.y = p2.y;
} else {
    UpCorner.x = p1.x;
    UpCorner.y = p1.y;
}
```

```

if ( p3.y > p4.y ) {
    DownCorner.x = p4.x;
    DownCorner.y = p4.y;
} else {
    DownCorner.x = p3.x;
    DownCorner.y = p3.y;
}
```

```

if ( p1.x > p4.x ) {
    LeftCorner.x = p1.x;
```



```

    LeftCorner.y = p1.y ;
} else {
    LeftCorner.x = p4.x ;
    LeftCorner.y = p4.y ;
}

if ( p2.x > p3.x ) {
    RightCorner.x = p3.x ;
    RightCorner.y = p3.y ;
} else {
    RightCorner.x = p2.x ;
    RightCorner.y = p2.y ;
}

*Xwidth = fabs(RightCorner.x - LeftCorner.x)/2.0 -2 ;
*Ywidth = fabs(UpCorner.y - DownCorner.y)/2.0 -2 ;

Center->x = min(UpCorner.x,DownCorner.x)+fabs(UpCorner.x-
DownCorner.x)/2.0 ;
Center->y = min(UpCorner.y,DownCorner.y)+fabs(UpCorner.y-
DownCorner.y)/2.0 ;

return 1 ;
}

// This function gives back the center of each tracking window in the prototype
// and the width and height of each window in the transformed model. All
// the information is in the Windows array.

int PASCAL get_tracking_windows(RPOINT *Prot,RPOINT *Model,
                               int From,int To,TR_WIN *Windows)
{
    double Xmax,Xmin ;
    double XModmax,XModmin ;
    double Xunit,XModunit ;
    double CurrX[20] ;
    RPOINT Cp1,Cp2,Cp3,Cp4 ;

```

```

RPOINT Cp[4] ;
RPOINT Mo[4] ;
int i ;
LINE UpLine,DownLine ;
LINE UpModLine,DownModLine ;
RPOINT p1,p2,p3,p4 ;
RPOINT mo1,mo2,mo3,mo4 ;
RPOINT DumCenterArr[20] ;

p1 = Prot[0] ;
p2 = Prot[1] ;
p3 = Prot[2] ;
p4 = Prot[3] ;
mo1 = Model[0] ;
mo2 = Model[1] ;
mo3 = Model[2] ;
mo4 = Model[3] ;
Xmax = max(p1.x,p4.x) ;
Xmin = min(p2.x,p3.x) ;
Xunit = (Xmin-Xmax)/(double)(To-From) ;
XModmax = max(mo1.x,mo4.x) ;
XModmin = min(mo2.x,mo3.x) ;
XModunit = (XModmin-XModmax)/(double)(To-From) ;
mkline(p1.x,p1.y,p2.x,p2.y,&UpLine) ;
mkline(p3.x,p3.y,p4.x,p4.y,&DownLine) ;
mkline(mo1.x,mo1.y,mo2.x,mo2.y,&UpModLine) ;
mkline(mo3.x,mo3.y,mo4.x,mo4.y,&DownModLine) ;

for ( i = From ; i < To-1 ; i++ ) {
    get_quad_segment_on_x(UpLine, DownLine,Xmax,Xunit,i,Cp) ;
    center_of_bounded_rect(Cp[0],Cp[1],Cp[2],Cp[3],&(Windows[i].Center),
                           &(Windows[i].Xdir),&(Windows[i].Ydir)) ;
    get_quad_segment_on_x(UpModLine,
DownModLine,XModmax,XModunit,i,Mo) ;
    center_of_bounded_rect(Mo[0],Mo[1],Mo[2],Mo[3],&(DumCenterArr[i]),
                           &(Windows[i].Xdir),&(Windows[i].Ydir)) ;
}

```

```

return 1 ;
}

```

```

// This function gets two horizontal (approx) lines, a start point on
// X axis a unit and a number of segment, and returns in Vertexes the
// vertexes of a rectangle in the number segment which is bounded inside
// these two lines.

```

```

int PASCAL get_quad_segment_on_x(LINE UpLine, LINE DownLine, double
XStart,

```

```

double Xunit, int SegNum, RPOINT *Vertexes)

```

```

{
double StartPoint, EndPoint ;

```

```

    StartPoint = XStart + Xunit*SegNum ;

```

```

    EndPoint = StartPoint + Xunit ;

```

```

    Vertexes[0].y = (UpLine.a*StartPoint + UpLine.c)/-UpLine.b ;

```

```

    Vertexes[0].x = StartPoint ;

```

```

    Vertexes[1].y = (UpLine.a*EndPoint + UpLine.c)/-UpLine.b ;

```

```

    Vertexes[1].x = EndPoint ;

```

```

    Vertexes[2].x = EndPoint ;

```

```

    Vertexes[2].y = (DownLine.a*EndPoint + DownLine.c)/-DownLine.b ;

```

```

    Vertexes[3].x = StartPoint ;

```

```

    Vertexes[3].y = (DownLine.a*StartPoint + DownLine.c)/-DownLine.b ;

```

```

return 1 ;
}

```

```

#include <windows.h>
#include <windowsx.h>
#include <math.h>
#include <commdlg.h>
#include <stdlib.h>
#include "const.h"
#include "bitmap.h"
#include "lines.h"
#include "track.h"
#include "persp.h"
#include "min_mag.h"
#include "lib.h"

```

```
#define SIZE 8
```

```

#define X_AXIS 1
#define Y_AXIS 2
#define MAXIMAL 1
#define MINIMAL 2

```

```

static double Fa[2 * FSIZE + 1], *F;
int PASCAL LSE_Perspective(HWND ,SHIFT_POINTS , SHIFT_POINTS
,int,
                                Perspective_Transform *) ;
int PASCAL xPerspEqn(double , double , double , double *) ;
int PASCAL yPerspEqn(double , double , double , double *) ;
int PASCAL qrsolv8 (HWND ,double m[SIZE][SIZE], int , double *) ;
int PASCAL qrdecomp (double m[SIZE][SIZE], int , double *, double *) ;
int PASCAL rsolv (double m[SIZE][SIZE], int , double *, double *) ;
int PASCAL invertPerspective(double Pf[3][3], double Pb[3][3]) ;
int PASCAL xThinEqn(double, double, double, double *) ;
int PASCAL yThinEqn(double, double, double, double *) ;
double PASCAL det3(double y[3][3]) ;
int PASCAL inv3(double y[3][3],double z[3][3]) ;
int PASCAL tmSim(SHIFT_POINTS , SHIFT_POINTS ,
int,Perspective_Transform *,
HFILE ) ;
int __cdecl dcomp(const void *a, const void *b) ;

```

366

```
double      norm2(RPOINT a, RPOINT b) ;
int PASCAL delete_externe_point(SHIFT_POINTS ,SHIFT_POINTS .
                                int , int ,int *);
```

```
MYBITMAP FAR * minify(HWND hwnd,MYBITMAP FAR* in, int fac)
{
MYBITMAP FAR *aux, *out;

/* Build windowed-sinc filter-table */
sinc_filter(fac);

/* horizontally sub-sample in -> aux */
aux = hminify(hwnd, in, fac);

/* vertically sub-sample Mid -> Out */
out = vminify(hwnd,aux, fac);

bm_free(aux);

return(out);
}
```

```
MYBITMAP FAR *hminify(HWND hwnd,MYBITMAP FAR *in, int fac)
{
MYBITMAP FAR *out;
int y;
int OutCols;
int ColorFactor ;

OutCols = in->cols / fac ;
out = bm_alloc(OutCols , in->rows, in->typ);

if(out == NULL)
    return(NULL);
if ( in->typ == COLOR_MODEL ) ColorFactor = 3 ;
else ColorFactor = 1 ;
```

```

out->gpic = (BYTE huge *)
    GlobalAllocPtr(GMEM_MOVEABLE,(DWORD)OutCols*(DWORD)(in-
>rows)*
                                (DWORD)ColorFactor);

```

```

if(in->typ == GREY_MODEL) {
    for(y = 0; y < in->rows; y++) {
        lpf1D(BITMAP_PLACE_PTR(in,y,0), in->cols, fac,
            BITMAP_PLACE_PTR(out,y,0) );
    }
}
else { // COLOR MODEL
    for(y = 0; y < in->rows; y++) {
        lpf1D_rgb(BITMAP_RGB_PLACE_PTR(in,y,0), in->cols, fac,
            BITMAP_RGB_PLACE_PTR(out,y,0) );
    }
}
return(out);
}

```

```

MYBITMAP FAR *vminify(HWND hwnd,MYBITMAP FAR *in, int fac)
{
    MYBITMAP FAR *out;
    int y, x;
    BYTE huge *ivec;
    BYTE huge *ovec;
    int OutRows ;
    int ColorFactor ;

```

```

    OutRows = in->rows / fac ;
    out = bm_alloc(in->cols, OutRows , in->typ);

```

```

    if(out == NULL)
        return(NULL);
    if ( in->typ == COLOR_MODEL ) ColorFactor = 3 ;
    else ColorFactor = 1 ;
    out->gpic = (BYTE huge *)

```

```

GlobalAllocPtr(GMEM_MOVEABLE,(DWORD)OutRows*(DWORD)(in->cols)*
                (DWORD)ColorFactor);
  ivec = (BYTE huge*)GlobalAllocPtr(GMEM_MOVEABLE,(DWORD)(in-
>rows)*
                (DWORD)ColorFactor);
  ovec = (BYTE huge *)GlobalAllocPtr(GMEM_MOVEABLE,(DWORD)(out-
>rows)*
                (DWORD)ColorFactor);

  if(in->typ == GREY_MODEL) {
    for(x = 0; x < in->cols; x++) {
      for(y = 0; y < in->rows; y++)
        ivec[y] = BITMAP_PLACE(in,y,x);
      lpf1D(ivec, in->rows, fac, ovec);
      for(y = 0; y < out->rows; y++)
        BITMAP_PLACE(out,y,x) = ovec[y];
    }
  }
  else {
    for(x = 0; x < in->cols; x++) {
      for(y = 0; y < in->rows; y++) {
        ivec[(DWORD)3*(DWORD)y] =
*(BITMAP_RGB_PLACE_PTR(in,y,x));
        ivec[(DWORD)3*(DWORD)y+1] =
*((BITMAP_RGB_PLACE_PTR(in,y,x))+1);
        ivec[(DWORD)3*(DWORD)y+2] =
*((BITMAP_RGB_PLACE_PTR(in,y,x))+2);
      }
      lpf1D_rgb(ivec, in->rows, fac, ovec);
      for(y = 0; y < out->rows; y++) {
        *(BITMAP_RGB_PLACE_PTR(out,y,x)) =
ovec[(DWORD)3*(DWORD)y];
        *((BITMAP_RGB_PLACE_PTR(out,y,x))+1) =
          ovec[(DWORD)3*(DWORD)y+1];
        *((BITMAP_RGB_PLACE_PTR(out,y,x))+2) =
          ovec[(DWORD)3*(DWORD)y+2];
      }
    }
  }

```

```

    }
}
return(out);
}

```

```

int PASCAL sinc_filter(int fac)
{
    int j;
    int fsize = LOBES * fac;
    double pie = acos(-1.0);
    double sum;

    F = Fa + fsize;

    for(j = -fsize; j <= fsize; j++) {
        F[j] = 1.0 / (double)(2 * fsize + 1);
    }

    /* Compute windowed sinc filter table */
    for(j = -fsize; j <= fsize; j++) {
        double x = (double)j / fac;
        double z, w;

        if(j == 0) {
            z = 1.0;
        }
        else {
            z = sin(pie * x) / (pie * x);
        }

        w = 0.5 + 0.5 * cos(pie * (double)j / (double)fsize);

        F[j] = z * w;
    }

    /* Normalize to unit integral */

```



```

    for(sum = 0, j = -fsize; j <= fsize; j++)
        sum += F[j];
    for(j = -fsize; j <= fsize; j++)
        F[j] /= sum;
    return 1;
}

int PASCAL lpf1D(BYTE huge* in, int Dim, int fac, BYTE huge *out)
{
    int i, j, k, x;
    int dim = Dim / fac;
    double acc;

    for(k = i = 0; i < dim; k += fac, i++) {

        acc = 0.0;

        for(j = -2 * fac; j <= 2 * fac; j++) {

            x = k + j;

            /* zero padding */
            if(x >= 0 && x < Dim)
                acc += (double)(*(in+x)) * F[j];
        }

        if(acc < 0.0)
            *(out+i) = 0;
        else if(acc > 255.0)
            *(out+i) = 255;
        else
            *(out+i) = (BYTE)(0.5 + acc);
    }
    return 1;
}

int PASCAL lpf1D_rgb(BYTE huge* in, int Dim, int fac, BYTE huge *out)
{

```

```

int i, j, k, x;
int dim = Dim / fac;
double accr, accg, accb;
DWORD   CurrPI;

for(k = i = 0; i < dim; k += fac, i++) {

    accr = accg = accb = 0.0;

    for(j = -2 * fac; j <= 2 * fac; j++) {

        x = k + j;

        /* zero padding */
        if(x >= 0 && x < Dim) {
            accr += (double)*(in+((DWORD)3*(DWORD)x))) * F[];
            accg += (double)*(in+((DWORD)3*(DWORD)x+1))) * F[];
            accb += (double)*(in+((DWORD)3*(DWORD)x+2))) * F[];
        }
    }

    CurrPI = (DWORD)3*(DWORD)i;
    if(accr < 0.0)
        *(out+CurrPI) = 0;
    else if(accr > 255.0)
        *(out+CurrPI) = 255;
    else
        *(out+CurrPI) = (BYTE)(0.5 + accr);

    CurrPI = (DWORD)3*(DWORD)i+1;
    if(accg < 0.0)
        *(out+CurrPI) = 0;
    else if(accg > 255.0)
        *(out+CurrPI) = 255;
    else
        *(out+CurrPI) = (BYTE)(0.5 + accg);

    CurrPI = (DWORD)3*(DWORD)i+2;
    373

```

```

        if(accb < 0.0)
            *(out+CurrPl) = 0;
        else if(accb > 255.0)
            *(out+CurrPl) = 255;
        else
            *(out+CurrPl) = (BYTE)(0.5 + accb);;
    }
    return 1 ;
}

```

```

int PASCAL edge_refine(MYBITMAP *Bmap,EDGE *e,int Len)
{
    double dx ;
    double dy ;
    int status;

    dx = fabs(e->xe - e->xs);
    dy = fabs(e->ye - e->ys);

    if(dx < dy)
        e->vertical = 1;
    else
        e->vertical = 0;

    if(e->vertical) {
        /* horizontal search for max. gradient */
        if(status = h_refine(Bmap,&(e->xs), &(e->ys),Len)) return(status);
        if(status = h_refine(Bmap,&(e->xe), &(e->ye),Len)) return(status);
    }
    else {
        /* vertical search for max. gradient */
        if(status = v_refine(Bmap,&(e->xs), &(e->ys),Len)) return(status);
        if(status = v_refine(Bmap,&(e->xe), &(e->ye),Len)) return(status);
    }
    return(0);
}

```

```

int PASCAL sub_pixel_interp(double ep, double ec, double en, double *zm,
                           double *em)
{
    /* 1D sup-pixel estimation of the registration error minimum:
    * ep = A (-1)^2 + B (-1) + C = A - B + C
    * ec = A ( 0)^2 + B ( 0) + C =      C
    * en = A (+1)^2 + B (+1) + C = A + B + C
    *
    * yields the following solution:
    */
    double C = ec;
    double A = (ep + en) / 2.0 - C;
    double B = (en - ep) / 2.0;
    double z;

    /* sup-pixel position estimate is zm */
    z = *zm = - (B / (2.0 * A));
    /*em = z*(A*z + B) + C ;
    *em = A * z * z + B * z + C;

    return(0);
}

```

```

#define EMAX 10000000L

```

```

#define R0 14 /* search range */
#define RY 6 /* Search range in Y direction*/
double Exy[2*RY+1][2*R0+1];
HFILE hFile;
OFSTRUCT of;

```

```

int PASCAL xysolve(HWND hwnd, MYBITMAP *CurrBmap, MYBITMAP
*DestBmap,

```

```

    SHIFT *Shifts, TR_WIN *Windows, int WindowsNum,
    TRACK_POINTS *TrBase, TRACK_POINTS *TrPoints,

```

```

        Perspective_Transform *NewTransf, HFILE hFile,
        TRACK_POINTS* DBasep, RPOINT *DiffArr)
{
    int k, num;
    SHIFT i;
    char String[50];
    int Wr = 0;
    int j;
    double HalfX, HalfY;
    SHIFT_POINTS InitPos, NewPos;
    RPOINT ShiftVals[NUM_OF_TRACK_POINTS];
    int Indexes[NUM_OF_TRACK_POINTS];
    int Size, Counter;
    int RemIndex;

    for ( k = 0 ; k < NUM_OF_TRACK_POINTS ; k++ ) {
        Shifts[k].dx = Shifts[k].dy = Shifts[k].sim = 0.0;
    }
    k = 0;
    Counter = 0;

    for(k = 0 ; k < TrPoints->NumOfPoints; k++) {
        //for(k = 0 ; k < WindowsNum; k++) {
            if ( xysrch(hwnd, CurrBmap, DestBmap, &(Shifts[k]),
                TrBase->TrackP[k], TrPoints->TrackP[k],
                //(int)(Windows[k].Xdir), (int)(Windows[k].Ydir), 10, 4) != 0 ) {
                CORR_WINDOWX, CORR_WINDOWY, /*R0, RY*/ 10, 5) != 0 )
            {
                ShiftVals[Counter].x = Shifts[k].dx;
                ShiftVals[Counter].y = Shifts[k].dy;
                Indexes[Counter] = k;
                Counter++;
            }
        }

        for ( k = 0 ; k < Counter ; k++ ) {
            //InitPos.TrackP[k].x = (double)DBasep->TrackP[Indexes[k]].x; //
        }
    }
    Replacing clustering.

```

```

        //InitPos.TrackP[k].y = (double)DBasep->TrackP[Indexes[k]].y;    //
Replacing clustering.
        InitPos.TrackP[k].x = (double)TrBase->TrackP[Indexes[k]].x;    //
Replacing clustering.
        InitPos.TrackP[k].y = (double)TrBase->TrackP[Indexes[k]].y;    //
Replacing clustering.
        NewPos.TrackP[k].x = TrPoints->TrackP[Indexes[k]].x +
        ShiftVals[k].x - DiffArr[Indexes[k]].x; // Replacing clustering.
        NewPos.TrackP[k].y = TrPoints->TrackP[Indexes[k]].y +
        ShiftVals[k].y - DiffArr[Indexes[k]].y;    // Replacing clustering.
    }
    Size = Counter;    // Replacing clustering.

```

```

strcpy(String,"InitPos After Cluster\n");    //
_lwrite(hFile,String,strlen(String));    //
for ( k = 0 ; k < Size ; k++ ) {    //
    sprintf(String,"%lf,%lf\n",InitPos.TrackP[k].x,InitPos.TrackP[k].y);//
    _lwrite(hFile,String,strlen(String));    //
}    //
strcpy(String,"\n\nNewPos After Cluster\n");    //
_lwrite(hFile,String,strlen(String));    //
for ( k = 0 ; k < Size ; k++ ) {    //
    sprintf(String,"%lf,%lf\n",NewPos.TrackP[k].x,NewPos.TrackP[k].y); //
    _lwrite(hFile,String,strlen(String));    //
}    //
_lwrite(hFile,"\n\n\n",3);
// Removing 20 % Of Bad Points
/*
delete_externe_point(InitPos,NewPos, Size, X_AXIS,&RemIndex) ;
for ( j = RemIndex ; j < Size-1 ; j++ ){
    InitPos.TrackP[j] = InitPos.TrackP[j+1];
    NewPos.TrackP[j] = NewPos.TrackP[j+1];
}
Size--;
delete_externe_point(InitPos,NewPos, Size, Y_AXIS,&RemIndex) ;
for ( j = RemIndex ; j < Size-1 ; j++ ){
    InitPos.TrackP[j] = InitPos.TrackP[j+1];

```

```

        NewPos.TrackP[j] = NewPos.TrackP[j+1];
    }
    Size--;
    /*
    if ( get_in_series_flag() == 1 ) {
        tmSim(InitPos, NewPos, Size, NewTransf, hFile);
    } else {
        tmSim(InitPos, NewPos, Size, NewTransf, hFile);
        //LSE_Perspective(hwnd, InitPos, NewPos, Size, NewTransf);
    }

    for ( k = 0; k < Size; k++ ) {
        double DstX, DstY, w;
        DstX = InitPos.TrackP[k].x * NewTransf->Pf[0][0] +
            InitPos.TrackP[k].y * NewTransf->Pf[1][0] + NewTransf-
>Pf[2][0];
        DstY = InitPos.TrackP[k].x * NewTransf->Pf[0][1] +
            InitPos.TrackP[k].y * NewTransf->Pf[1][1] + NewTransf-
>Pf[2][1];
        w = InitPos.TrackP[k].x * NewTransf->Pf[0][2] +
            InitPos.TrackP[k].y * NewTransf->Pf[1][2] + NewTransf-
>Pf[2][2];
        sprintf(String, "%lf->%lf::%lf->%lf\n", InitPos.TrackP[k].x, DstX/w,
            InitPos.TrackP[k].y, DstY/w);
        _lwrite(hFile, String, strlen(String));
    }
    for ( j = 0; j < 3; j++ ) {
        //
        sprintf(String, "\n%lf, %lf, %lf\n", NewTransf->Pf[j][0], //
            NewTransf->Pf[j][1], NewTransf->Pf[j][2]); //
        _lwrite(hFile, String, strlen(String)); //
    }
    //
    return(1);
}

```

```

int PASCAL xysrch(HWND hwnd, MYBITMAP *CurrBmap, MYBITMAP
*OriginSign,

```

```

SHIFT *s, POINT pBase, POINT pLast, int CorWinX, int CorWinY, int XWin, int
YWin)
{
int x_0, y_0;
int dx, dy, k, l;
int x_n, y_n;
int dj, di;
double t, r;
int i;
double em; double e;
char String[50];
int long PixInWin;
char Buffer[256];
long Tresh;
BYTE huge *CurrPtr;
BYTE huge *ZeroPtr;
int FromX, FromY;
double z, d, area;

```

```

x_0 = pBase.x;
y_0 = pBase.y;

```

```

FromX = pLast.x;
FromY = pLast.y;

```

```

PixInWin = (DWORD)(CorWinX*2+1)*(DWORD)(CorWinY*2+1);
Tresh = (DWORD)12*(DWORD)PixInWin;
//Tresh = (DWORD)4*(DWORD)PixInWin;

```

```

area = (2 * CorWinY + 1) * (2 * CorWinX + 1);
em = 1.0e20;
y_n = FromY - YWin - 1;
for(dy = -YWin; dy <= YWin; dy++) {
    y_n++;
    for(dx = -XWin; dx <= XWin; dx++) {
        x_n = FromX + dx;
        e = 0.0;
    }
}

```



```

    for(k = -CorWinY; k <= CorWinY; k++) {
        CurrPtr = BITMAP_PLACE_PTR(CurrBmap,y_n+k,x_n) ;
        ZeroPtr = BITMAP_PLACE_PTR(OriginSign,y_0+k,x_0) ;
        for(l = -CorWinX; l <= CorWinX; l++) {
            z = CurrPtr[l] - ZeroPtr[l] ;
            e = e + z * z;
        }
    }

    e /= area;
    Exy[dy+YWin][dx+XWin] = e;

    /* update min. error */
    if(e < em) {
        em = e;
        dj = dx;
        di = dy;
    }
}
}
s->dx = dj ;
s->dy = di ;
if ( em > Tresh ) return 0 ;
if ( abs(dj) == XWin || abs(di) == YWin ) return 0 ;
sub_pixel_refine(dj,di,dj+XWin, di+YWin, s);

return 1 ;
}

int PASCAL sub_pixel_refine(int dj, int di, int dxIndex,int dyIndex,SHIFT *s)
{
    double deltax, deltax;
    double simx, simy;

    s->dx = dj;
    s->dy = di;

    s->sim = Exy[dyIndex][dxIndex];

```

```

sub_pixel_interp(Exy[dylndex][dxIndex-1], Exy[dylndex][dxIndex],
                  Exy[dylndex][dxIndex+1], &deltax, &simx);
sub_pixel_interp(Exy[dylndex-1][dxIndex], Exy[dylndex][dxIndex],
                  Exy[dylndex+1][dxIndex], &deltay, &simy);

s->dx += deltax;
s->dy += deltay;

s->sim = (simx + simy) / 2.0;

return(0);
}

```

```

static double Mp[8][8], Ap[8];
double G[2 * NUM_OF_TRACK_POINTS][8];
double Gt[8][2 * NUM_OF_TRACK_POINTS];
double Fpr[2 * NUM_OF_TRACK_POINTS];

```

```

int PASCAL Quad2Quad(HWND hwnd, RPOINT srcpnts[4], RPOINT
dstpnts[4],

```

Perspective\_Transform \*Tp)

```

{
int i,j, k, status;
double x,y,u,v ;

```

```

for(k = 0; k < 4; k++) {
    x = dstpnts[k].x;
    y = dstpnts[k].y;
    u = srcpnts[k].x;
    v = srcpnts[k].y;

```

```

    Ap[ k] = x;

```

```

    Ap[4 + k] = y;

    xPerspEqn(x, u, v, Mp[ k]);
    yPerspEqn(y, u, v, Mp[4 + k]);
}
qrsolv8(hwnd, Mp, 8, Ap);

Tp->Pf[0][0] = Ap[0];
Tp->Pf[1][0] = Ap[1];
Tp->Pf[2][0] = Ap[2];
Tp->Pf[0][1] = Ap[3];
Tp->Pf[1][1] = Ap[4];
Tp->Pf[2][1] = Ap[5];
Tp->Pf[0][2] = Ap[6];
Tp->Pf[1][2] = Ap[7];
Tp->Pf[2][2] = 1.0;

status = invertPerspective(Tp->Pf, Tp->Pb);

return(1);

}

int PASCAL copy_transform(Perspective_Transform
*To, Perspective_Transform *From)
{
    int i, j;

    for (i = 0; i < 3; i++) {
        for (j = 0; j < 3; j++) {
            To->Pf[i][j] = From->Pf[i][j];
            To->Pb[i][j] = From->Pb[i][j];
        }
    }

    return 1;
}

```

```

int PASCAL LSE_Perspective(HWND hwnd,SHIFT_POINTS src_pts,
    SHIFT_POINTS dst_pts, int pnum,Perspective_Transform *Tp)
{
    int i,j, k, status;
    double x,y,u,v ;

    if(pnum < 4) {
        MessageBox (hwnd, "Cannot solve perspective with less than 3
points."
        , "Shifts", MB_ICONEXCLAMATION | MB_OK) ;
        return 0 ;
    }

    if(pnum == 4) { /* Quad2Quad */
        for(k = 0; k < 4; k++) {
            x = dst_pts.TrackP[k].x;
            y = dst_pts.TrackP[k].y;
            u = src_pts.TrackP[k].x;
            v = src_pts.TrackP[k].y;

            Ap[ k] = x;
            Ap[4 + k] = y;

            xPerspEqn(x, u, v, Mp[ k]);
            yPerspEqn(y, u, v, Mp[4 + k]);
        }
    }

    else {
        for(k = 0; k < pnum; k++) {
            x = dst_pts.TrackP[k].x;
            y = dst_pts.TrackP[k].y;
            u = src_pts.TrackP[k].x;
            v = src_pts.TrackP[k].y;

            Fpr[ k] = x;
            Fpr[pnum + k] = y;
        }
    }
}

```

```

    xPerspEqn(x, u, v, G[k]);
    yPerspEqn(y, u, v, G[pnum + k]);
}

for(k = 0; k < 2 * pnum; k++) {
    for(j = 0; j < 8; j++) {
        Gt[j][k] = G[k][j];
    }
}

for (i = 0; i < 8; i++) {
    for (j = 0; j < 8; j++) {
        Mp[i][j] = 0.0;
        for (k = 0; k < 2 * pnum; k++) {
            Mp[i][j] += Gt[i][k] * G[k][j];
        }
    }
}

for(j = 0; j < 8; j++) {
    Ap[j] = 0;
    for(k = 0; k < pnum * 2; k++) {
        Ap[j] += Gt[j][k] * Fpr[k];
    }
}

qrsolv8(hwnd, Mp, 8, Ap);

Tp->Pf[0][0] = Ap[0];
Tp->Pf[1][0] = Ap[1];
Tp->Pf[2][0] = Ap[2];
Tp->Pf[0][1] = Ap[3];
Tp->Pf[1][1] = Ap[4];
Tp->Pf[2][1] = Ap[5];
Tp->Pf[0][2] = Ap[6];
Tp->Pf[1][2] = Ap[7];

```

```

    Tp->Pf[2][2] = 1.0;

    status = invertPerspective(Tp->Pf, Tp->Pb);

    return(1);
}

int PASCAL xPerspEqn(double x, double u, double v, double *xRow)
{
    xRow[0] = u;
    xRow[1] = v;
    xRow[2] = 1;
    xRow[3] = 0;
    xRow[4] = 0;
    xRow[5] = 0;
    xRow[6] = - u * x;
    xRow[7] = - v * x;
}

int PASCAL yPerspEqn(double y, double u, double v, double *yRow)
{
    yRow[0] = 0;
    yRow[1] = 0;
    yRow[2] = 0;
    yRow[3] = u;
    yRow[4] = v;
    yRow[5] = 1;
    yRow[6] = - u * y;
    yRow[7] = - v * y;
}

int PASCAL qrsolv8 (HWND hwnd, double m[SIZE][SIZE], int size, double
b[SIZE])
{
    int    i,
           j;

```

```

double      tau,
      m1[SIZE],
      m2[SIZE];

if (qrdecomp (m, size, m1, m2) < 0) {
    MessageBox (hwnd, "singularity in qrdecomp()."
        , "Shifts", MB_ICONEXCLAMATION | MB_OK);
    return 0;
}

for (j = 0; j < (size - 1); j++) {
    tau = 0.0;
    for (i = j; i < size; i++)
        tau += m[i][j] * b[i];
    tau /= m1[j];
    for (i = j; i < size; i++)
        b[i] -= tau * m[i][j];
}
b[size - 1] /= m2[size - 1];
rsolv (m, size, m2, b);
}
/*
*      Compute the QR decomposition of a square matrix m using the
Stewart
*      algorithm.
*      Upon termination , thr decomposition is stored in m,m1 and m2 as
*      follows:
*      R is contained in the upper triangle of m except that its main
*      diagonal is contained in m2, and  $Q(transpos)=Q(n-1)*...*Q(1)$ 
*      where  $Q(j)=I-(U_j*U_j(transpos)/P_j)$  where
*       $U_j[i]=0, i=1 \rightarrow j-1, U_j[i]=m[i][j], i=j \rightarrow n, P_j=m1[j]$ .
*
*      Stewart,G.W., Introduction to matrix Computations , Academic Press,
*      New York (1973).
*
* C Implementaion: Dr. I. Wilf.
*/

```

```

int PASCAL qrdecomp (double m[SIZE][SIZE], int size, double m1[SIZE],
double m2[SIZE])
{
    int    i,k,j;
    double    eta,t,sigma,tau ;

    for (k = 0; k < (size - 1); k++) {
        eta = 0.0;
        for (i = k; i < size; i++)
            if (fabs (m[i][k]) > eta)
                eta = fabs (m[i][k]);
        if (eta == 0.0)
            return (-1);
        /* form Qk and premultiply m by it */
        t = 0;
        for (i = k; i < size; i++) {
            m[i][k] /= eta;
            t += m[i][k] * m[i][k];
        }
        if(m[k][k] >= 0.0)
            sigma = sqrt(t);
        else
            sigma = -sqrt(t);
        m[k][k] += sigma;
        m1[k] = sigma * m[k][k];
        m2[k] = (-eta * sigma);
        tau = 0;
        for (j = k + 1; j < size; j++) {
            tau = 0;
            for (i = k; i < size; i++)
                tau += m[i][k] * m[i][j];
            tau /= m1[k];
            for (i = k; i < size; i++)
                m[i][j] -= tau * m[i][k];
        }
    }
    m2[size - 1] = m[size - 1][size - 1];
    return (0);
}

```



```

}
/*
 *   rsolv(m,size,m2,b)
 *   solve Rx=b for b, where the upper triangular matrix R is
 *   stored in M , m2.
 *
 * C Implementaion: Dr. I. Wilf.
 */

```

```

int PASCAL rsolv (double m[SIZE][SIZE], int size, double *m2, double *b)
{
    int    i,j;
    double    s;

    for (i = size - 2; i >= 0; i--) {
        s = 0;
        for (j = i + 1; j < size; j++)
            s += m[i][j] * b[j];
        b[i] = (b[i] - s) / m2[i];
    }
}

```

```

int PASCAL invertPerspective(double Pf[3][3], double Pb[3][3])
{

```

```

    double a11 = Pf[0][0];
    double a12 = Pf[0][1];
    double a13 = Pf[0][2];
    double a21 = Pf[1][0];
    double a22 = Pf[1][1];
    double a23 = Pf[1][2];
    double a31 = Pf[2][0];
    double a32 = Pf[2][1];
    double a33 = Pf[2][2];

```

```

    Pb[0][0] = a22 * a33 - a23 * a32;
    Pb[0][1] = a13 * a32 - a12 * a33;
    Pb[0][2] = a12 * a23 - a13 * a22;

```

```

Pb[1][0] = a23 * a31 - a21 * a33;
Pb[1][1] = a11 * a33 - a13 * a31;
Pb[1][2] = a13 * a21 - a11 * a23;
Pb[2][0] = a21 * a32 - a22 * a31;
Pb[2][1] = a12 * a31 - a11 * a32;
Pb[2][2] = a11 * a22 - a12 * a21;

```

```

inv3(Pf, Pb);

```

```

return(0);

```

```

}

```

```

int PASCAL inv3(double y[3][3], double z[3][3])

```

```

{

```

```

double b[2][2], dety, detb;

```

```

int i, j, k, l, k1, l1;

```

```

dety=det3(y);

```

```

if(dety==0.0) return(1);

```

```

/* compute [] */

```

```

for(i=0; i< 3; i++)

```

```

for(j=0; j< 3; j++) {

```

```

for(k = 0; k < 2; k++)

```

```

for(l = 0; l < 2; l++) {

```

```

if(l<i) l1=l;

```

```

else l1=l+1;

```

```

if(k<j) k1=k;

```

```

else k1=k+1;

```

```

b[k][l]=y[k1][l1];

```

```

}

```

```

detb=b[0][0]*b[1][1]-b[1][0]*b[0][1];

```

```

if(((i+j)%2)==0) z[i][j]=detb/dety;

```

```

else z[i][j]=(-detb)/dety;

```

```

}

```

```

}

```

```

double PASCAL det3(double y[3][3])

```

```

{
short j,k,l;
double b[2][2],det,detb;

    det=0.0;
    for(j=0;j<=2;j++)
    {
        for(k=0;k<=1;k++)
        for(l=0;l<=1;l++)
        {
            if(l<j) b[k][l]=y[k+1][l];
            else b[k][l]=y[k+1][l+1];
        }
        detb=b[0][0]*b[1][1]-b[0][1]*b[1][0];
        if((j%2)==0) det=det+y[0][j]*detb;
        else det=det-y[0][j]*detb;
    }
    return(det);
}

/*
int PASCAL invertPerspective(double Pf[3][3], double Pb[3][3])
{
double a11,a12,a13,a21,a22,a23,a31,a32,a33 ;

    a11 = Pf[0][0];
    a12 = Pf[0][1];
    a13 = Pf[0][2];
    a21 = Pf[1][0];
    a22 = Pf[1][1];
    a23 = Pf[1][2];
    a31 = Pf[2][0];
    a32 = Pf[2][1];
    a33 = Pf[2][2];

    Pb[0][0] = a22 * a33 - a23 * a32;
    Pb[0][1] = a13 * a32 - a12 * a33;
    Pb[0][2] = a12 * a23 - a13 * a22;

```

```

    Pb[1][0] = a23 * a31 - a21 * a33;
    Pb[1][1] = a11 * a33 - a13 * a31;
    Pb[1][2] = a13 * a21 - a11 * a23;
    Pb[2][0] = a21 * a32 - a22 * a31;
    Pb[2][1] = a12 * a31 - a11 * a32;
    Pb[2][2] = a11 * a22 - a12 * a21;

    return(0);
}
*/

int PASCAL h_refine(MYBITMAP *Bmap, double *x, double *y, int Len)
{
    int i, j, k, km;
    int z = Len + 1;
    double u, gm;
    double gmax;
    double a, b, c, d, e, f;
    double Rx, Gx, Bx;
    int rows, cols;
    DWORD Size;
    int EfectLen;
    double *Gh;

    i = (int)(*y + 0.5);
    j = (int)(*x + 0.5);
    cols = Bmap->cols;
    rows = Bmap->rows;

    if(j < z || j > (cols - z)) return(1);
    if(i < 1 || i > (rows - 2)) return(1);

    EfectLen = (DWORD)Len*(DWORD)2+1;
    Size = (DWORD)(sizeof(double))*(DWORD)Len*(DWORD)2+2;
    Gh = (double *)GlobalAllocPtr(GMEM_MOVEABLE, Size);

    for(k = -Len; k <= Len; k++) {
        a = *(BITMAP_RGB_PLACE_PTR(Bmap, i-1, j+k-1));

```

```

b = *(BITMAP_RGB_PLACE_PTR(Bmap,i,j+k-1));
c = *(BITMAP_RGB_PLACE_PTR(Bmap,i+1,j+k-1));
d = *(BITMAP_RGB_PLACE_PTR(Bmap,i-1,j+k));
e = *(BITMAP_RGB_PLACE_PTR(Bmap,i,j+k));
f = *(BITMAP_RGB_PLACE_PTR(Bmap,i+1,j+k));

```

```

Rx = a + 2.0 * b + c - d - 2.0 * e - f;

```

```

a = *(BITMAP_RGB_PLACE_PTR(Bmap,i-1,j+k-1)+1);
b = *(BITMAP_RGB_PLACE_PTR(Bmap,i,j+k-1)+1);
c = *(BITMAP_RGB_PLACE_PTR(Bmap,i+1,j+k-1)+1);
d = *(BITMAP_RGB_PLACE_PTR(Bmap,i-1,j+k)+1);
e = *(BITMAP_RGB_PLACE_PTR(Bmap,i,j+k)+1);
f = *(BITMAP_RGB_PLACE_PTR(Bmap,i+1,j+k)+1);
Gx = a + 2.0 * b + c - d - 2.0 * e - f;

```

```

a = *(BITMAP_RGB_PLACE_PTR(Bmap,i-1,j+k-1)+2);
b = *(BITMAP_RGB_PLACE_PTR(Bmap,i,j+k-1)+2);
c = *(BITMAP_RGB_PLACE_PTR(Bmap,i+1,j+k-1)+2);
d = *(BITMAP_RGB_PLACE_PTR(Bmap,i-1,j+k)+2);
e = *(BITMAP_RGB_PLACE_PTR(Bmap,i,j+k)+2);
f = *(BITMAP_RGB_PLACE_PTR(Bmap,i+1,j+k)+2);
Bx = a + 2.0 * b + c - d - 2.0 * e - f;

```

```

Gh[k+Len] = sqrt(Rx * Rx + Gx * Gx + Bx * Bx);

```

```

}

```

```

/* Find maximal response */

```

```

km = 0;

```

```

gm = Gh[0];

```

```

for(k = 1 ; k < EfectLen; k++) {

```

```

    if(Gh[k] > gm) {

```

```

        km = k;

```

```

        gm = Gh[k];

```

```

    }

```

```

}

```

```

if(km == 0 || km == EfectLen) {

```

```

        GlobalFreePtr(Gh) ;
        return(1);
    }
    sub_pixel_interp(Gh[km-1], Gh[km], Gh[km+1], &u, &gmax);
    /* can threshold gmax to decide if edge detection was successful */
    GlobalFreePtr(Gh) ;
    if ( gmax < 100.0 ) return(1) ;
    *x += u+km;

    return(0);
}

int PASCAL v_refine(MYBITMAP *Bmap,double *x, double *y,int Len)
{
    int i, j, k, km;
    int z = Len + 1;
    double v, gm;
    double gmax;
    int cols,rows ;
    DWORD Size ;
    int EfectLen ;
    double a, b, c, d, e, f;
    double Ry, Gy, By;
    double *Gh ;

    i = (int)(*y + 0.5);
    j = (int)(*x + 0.5);
    cols = Bmap->cols ;
    rows = Bmap->rows ;
    if(j < z || j > (cols - z)) return(1);
    if(i < 1 || i > (rows - 2)) return(1);

    EfectLen = (DWORD)Len*(DWORD)2+1 ;
    Size = (DWORD)(sizeof(double))*(DWORD)Len*(DWORD)2+2 ;
    Gh = (double *)GlobalAllocPtr(GMEM_MOVEABLE,Size) ;

    for(k = -Len; k <= Len; k++) {
        a = *(BITMAP_RGB_PLACE_PTR(Bmap,i+k-1,j-1)) ;

```

```

b = *(BITMAP_RGB_PLACE_PTR(Bmap,i+k-1,j));
c = *(BITMAP_RGB_PLACE_PTR(Bmap,i+k-1,j+1));
d = *(BITMAP_RGB_PLACE_PTR(Bmap,i+k,j-1));
e = *(BITMAP_RGB_PLACE_PTR(Bmap,i+k,j));
f = *(BITMAP_RGB_PLACE_PTR(Bmap,i+k,j+1));

```

```

Ry = a + 2.0 * b + c - d - 2.0 * e - f;

```

```

a = *(BITMAP_RGB_PLACE_PTR(Bmap,i+k-1,j-1)+1);
b = *(BITMAP_RGB_PLACE_PTR(Bmap,i+k-1,j)+1);
c = *(BITMAP_RGB_PLACE_PTR(Bmap,i+k-1,j+1)+1);
d = *(BITMAP_RGB_PLACE_PTR(Bmap,i+k,j-1)+1);
e = *(BITMAP_RGB_PLACE_PTR(Bmap,i+k,j)+1);
f = *(BITMAP_RGB_PLACE_PTR(Bmap,i+k,j+1)+1);

```

```

Gy = a + 2.0 * b + c - d - 2.0 * e - f;

```

```

a = *(BITMAP_RGB_PLACE_PTR(Bmap,i+k-1,j-1)+2);
b = *(BITMAP_RGB_PLACE_PTR(Bmap,i+k-1,j)+2);
c = *(BITMAP_RGB_PLACE_PTR(Bmap,i+k-1,j+1)+2);
d = *(BITMAP_RGB_PLACE_PTR(Bmap,i+k,j-1)+2);
e = *(BITMAP_RGB_PLACE_PTR(Bmap,i+k,j)+2);
f = *(BITMAP_RGB_PLACE_PTR(Bmap,i+k,j+1)+2);
By = a + 2.0 * b + c - d - 2.0 * e - f;

```

```

Gh[k+Len] = sqrt(Ry * Ry + Gy * Gy + By * By);

```

```

}

```

```

/* Find maximal response */

```

```

km = 0;

```

```

gm = Gh[0];

```

```

for(k = 1; k < EfectLen; k++) {

```

```

    if(Gh[k] > gm) {

```

```

        km = k;

```

```

        gm = Gh[k];

```

```

    }

```

```

}

```

```

if(km == 0 || km == EfectLen) {

```

```

    GlobalFreePtr(Gh);

```

```

        return(1);
    }
    sub_pixel_interp(Gh[km-1], Gh[km], Gh[km+1], &v, &gmax);
    /* can threshold gmax to decide if edge detection was successful */
    GlobalFreePtr(Gh);
    if ( gmax < 100.0 ) return(1);
    *y += v+km;
    return(0);
}

#define ORDER 8
#define NDATA 20
static double qrM[ORDER][ORDER], qrA[ORDER];

double qrG[2 * NDATA][ORDER];
double qrGt[ORDER][2 * NDATA];
double qrF[2 * NDATA];

int PASCAL Thin_Perspective(HWND hwnd, SHIFT_POINTS src_pts,
    SHIFT_POINTS dst_pts, int pnum, Perspective_Transform *Tp)
{
    int i, j, k, status;

    if(pnum < 3) {
        MessageBox (hwnd, "Cannot solve perspective with less than 3
points."
        , "Shifts", MB_ICONEXCLAMATION | MB_OK);
        return 0;
    }

    for(k = 0; k < pnum; k++) {
        double x = dst_pts.TrackP[k].x;
        double y = dst_pts.TrackP[k].y;
        double u = src_pts.TrackP[k].x;
        double v = src_pts.TrackP[k].y;

        qrF[    k] = x;

```



```

qrF[pnum + k] = y;

xThinEqn(x, u, v, qrG[k]);
yThinEqn(y, u, v, qrG[pnum + k]);
continue;
}

```

```

for(k = 0; k < 2 * pnum; k++) {
    for(j = 0; j < 5; j++) {
        qrGt[j][k] = qrG[k][j];
    }
}

```

```

for(i = 0; i < 5; i++)
for(j = 0; j < 5; j++) {
    qrM[i][j] = 0.0;
    for(k = 0; k < 2 * pnum; k++) {
        qrM[i][j] += qrGt[i][k] * qrG[k][j];
    }
}

```

```

for(j = 0; j < 5; j++) {
    qrA[j] = 0;
    for(k = 0; k < pnum * 2; k++) {
        qrA[j] += qrGt[j][k] * qrF[k];
    }
}

```

```

qrsolv(qrM, 5, qrA);

```

```

Tp->Pf[0][0] = qrA[0];
Tp->Pf[1][0] = qrA[1];
Tp->Pf[2][0] = qrA[2];
Tp->Pf[0][1] = -qrA[1];
Tp->Pf[1][1] = qrA[0];
Tp->Pf[2][1] = qrA[3];
Tp->Pf[0][2] = qrA[4];
Tp->Pf[1][2] = 0.0;

```

```

    Tp->Pf[2][2] = 1.0;

    status = invertPerspective(Tp->Pf, Tp->Pb);

    return(1);
}

int PASCAL xThinEqn(double x, double u, double v, double *xRow)
{
    xRow[0] = u;
    xRow[1] = v;
    xRow[2] = 1;
    xRow[3] = 0;
    xRow[4] = - u * x;
    return 1 ;
}

int PASCAL yThinEqn(double y, double u, double v, double *yRow)
{
    yRow[0] = v;
    yRow[1] = -u;
    yRow[2] = 0;
    yRow[3] = 1;
    yRow[4] = - u * y;
    return 1 ;
}

int PASCAL trans_grey_frame_to_fields(HWND hwnd,RPOINT
*SrcPnts,RPOINT *DstPnts,
    MYBITMAP *SrcBmap,RPOINT *FrPnts, MYBITMAP **F1,MYBITMAP
**F2)
{
    Perspective_Transform TpCurr,BasicTransf ;
    RPOINT    FrDst[4],CmPoly[4] ;
    RECT Rectan ;
    MYBITMAP *DestBmap ;
    DWORD Size ;
    int Cols,Rows ;

```

```

Quad2Quad(hwnd,FrPnts,DstPnts,&TpCurr);
l_find_bound_rect(DstPnts, &Rectan);
l_quad_in_new_origin(CrmPoly,DstPnts,Rectan.left,Rectan.top,4);
Cols = Rectan.right-Rectan.left+1;
Rows = Rectan.bottom-Rectan.top+1;
*F1 = bm_alloc(Cols,Rows,GREY_MODEL);
Size = (DWORD)Cols*(DWORD)Rows;
(*F1)->gpic = (BYTE huge*)GlobalAllocPtr(GMEM_MOVEABLE,Size);
perspective(hwnd,SrcBmap,*F1,FrPnts,
            CrmPoly, GREY_MODEL, &BasicTransf);

```

```

//split_bitmap_frame(SrcBmap,F1, F2);
//bm_free(DestBmap);

```

```

return 1;

```

```

}

```

```

#define NZOOM 256

```

```

#define WNUM 32

```

```

double rZooms[NZOOM];
double rTheta[NZOOM];
double rDx[WNUM];
double rDy[WNUM];
double ValArr[] = {0.0, 0.0, 0.0, 0.0};
double Wheight[] = {2.0,2.0,3.0,3.0};

```

```

int PASCAL tmSim(SHIFT_POINTS mpts, SHIFT_POINTS fpts, int pnun,
                Perspective_Transform *Tp, HFILE hFile)

```

```

{

```

```

    double zoom, theta;
    double zsin, zcos;
    char String[50];
    double dx, dy;
    double Orgtheta;

```

```

    int i, j, nZoom = 0;
    double mlen, flen;

```

```

double mtheta, ftheta ;

if(pnum > WNUM)
    pnum = WNUM;

for(i = 0; i < pnum; i++)
    for(j = i + 1; j < pnum; j++) {
        mlen = norm2(mpts.TrackP[i], mpts.TrackP[j]);
        if(mlen < 30.0)
            continue;

        flen = norm2(fpts.TrackP[i], fpts.TrackP[j]);

        mtheta = atan2(mpts.TrackP[j].x - mpts.TrackP[i].x,
                       mpts.TrackP[j].y - mpts.TrackP[i].y);
        ftheta = atan2(fpts.TrackP[j].x - fpts.TrackP[i].x,
                       fpts.TrackP[j].y - fpts.TrackP[i].y);

        zoom = flen / mlen;
        theta = ftheta - mtheta;

        if(nZoom < NZOOM) {
            rZooms[nZoom] = zoom;
            rTheta[nZoom] = theta;
            ++nZoom;
        }
    }

{ /* Trimmed-mean estimate of theta */
    double tavg = 0.0;
    int q1 = nZoom / 4;
    int q3 = 3 * q1;

    qsort((void*)rTheta, (size_t)nZoom, sizeof(double), dcomp);
    for(i = q1; i < q3; i++)
        tavg += rTheta[i];
    tavg /= (double)(q3 - q1);
}

```

```

    theta = tavg;
}
/* Trimmed-mean estimate of zoom */
double zavg = 0.0;
int q1 = nZoom / 4;
int q3 = 3 * q1;

qsort((void*) rZooms, (size_t) nZoom, sizeof(double), dcomp);
for(i = q1; i < q3; i++)
    zavg += rZooms[i];
zavg /= (double)(q3 - q1);
zoom = zavg;
}

```

```

//Orgtheta = theta ;
//smooth_values(Orgtheta, &theta, ValArr, 4, Wheight) ;
sprintf(String, "\n%ZOOM %lf, THETA %lf\n", zoom, theta) ; //
_wwrite(hFile, String, strlen(String)) ;
zcos = zoom * cos(theta);
zsin = zoom * sin(theta);

```

```

for(i = 0; i < pnum; i++) {
    rDx[i] = fpts.TrackP[i].x - zcos * mpts.TrackP[i].x -
              zsin * mpts.TrackP[i].y;
    rDy[i] = fpts.TrackP[i].y + zsin * mpts.TrackP[i].x -
              zcos * mpts.TrackP[i].y;
}

```

```

/* Trimmed-mean estimate of dx */
double xavg = 0.0;
int q1 = pnum / 4;
int q3 = 3 * q1;

qsort((void*) rDx, (size_t) pnum, sizeof(double), dcomp);
for(i = q1; i < q3; i++)
    xavg += rDx[i];
xavg /= (double)(q3 - q1);
dx = xavg;

```

```

    }

    { /* Trimmed-mean estimate of dy */
        double yavg = 0.0;
        int q1 = pnum / 4;
        int q3 = 3 * q1;

        qsort((void*)rDy, (size_t)pnum, sizeof(double), dcomp);
        for(i = q1; i < q3; i++)
            yavg += rDy[i];
        yavg /= (double)(q3 - q1);
        dy = yavg;
    }

    { /* Fill fwd. matrix */

        Tp->Pf[0][0] = zcos;
        Tp->Pf[1][0] = zsin;
        Tp->Pf[2][0] = dx;

        Tp->Pf[0][1] = -zsin;
        Tp->Pf[1][1] = zcos;
        Tp->Pf[2][1] = dy;

        Tp->Pf[0][2] = 0;
        Tp->Pf[1][2] = 0;
        Tp->Pf[2][2] = 1;

        inv3(Tp->Pf, Tp->Pb);
    }
    return(1);
}

```

```

double    norm2(RPOINT a, RPOINT b)
{
    double dx = a.x - b.x;
    double dy = a.y - b.y;

```

```

    return(sqrt(dx * dx + dy * dy));
}

int __cdecl dcomp(const void *a, const void *b)/(double *a, double *b)/
{
    if(*(double*)a == *(double*)b) return(0);
    return(*(double*)a < *(double*)b ? -1 : 1);
}

int PASCAL delete_externe_point(SHIFT_POINTS init,SHIFT_POINTS new,
                                int Size, int Axis,int *Index)
{
    int i ;
    double Value,Tmp ;

    if (Axis = X_AXIS ) {
        Value = -100.0 ;
        for ( i = 0 ; i < Size ; i++ ) {
            if ( (Tmp = fabs(new.TrackP[i].x - init.TrackP[i].x)) > Value ) {
                Value = Tmp ;
                *Index = i ;
            }
        }
    }

    if (Axis = Y_AXIS ) {
        Value = -100.0 ;
        for ( i = 0 ; i < Size ; i++ ) {
            if ( (Tmp = fabs(new.TrackP[i].y - init.TrackP[i].y)) > Value ) {
                Value = Tmp ;
                *Index = i ;
            }
        }
    }

    return 1 ;
}

```

## C L A I M S

1  
2  
3  
4  
5 1. Apparatus for advertisement site detection  
6 comprising:

7 a field grabber operative to grab and  
8 digitize at least one field representing at least a  
9 portion of a sports facility; and

10 an advertisement site detector operative to  
11 detect at least one advertisement site in at least one  
12 field on a basis other than location of the  
13 advertisement site relative to the sports facility.  
14

15 2. Apparatus for advertisement incorporation  
16 comprising:

17 a field grabber operative to grab and  
18 digitize at least one field representing at least a  
19 portion of a sports facility; and

20 an advertisement incorporator operative to  
21 incorporate a portion of an advertisement into at least  
22 one field at a partially occluded advertisement site  
23 within the sports facility, wherein the portion of the  
24 incorporated advertisement corresponds to an unoccluded  
25 portion of the advertisement site.  
26

27 3. Apparatus according to claim 2 wherein said  
28 advertisement incorporator includes an advertisement  
29 site detector operative to detect at least one  
30 advertisement site in at least one field on a basis  
31 other than location of the advertisement site relative  
32 to the sports facility.  
33

34 4. Apparatus for advertisement site detection  
35 comprising:

36 a field grabber operative to grab and  
37 digitize at least one field representing at least a  
38 portion of a sports facility;



1  
2 an advertisement site detector operative to  
3 detect at least one advertisement site in at least one  
4 field; and

5 an advertisement site tracker operative to  
6 track at least one advertisement site detected by the  
7 advertisement site detector without tracking an entire  
8 field.

9  
10 5. Apparatus according to claim 2 wherein said  
11 advertisement incorporator includes:

12 an advertisement site detector operative to  
13 detect at least one advertisement site in at least one  
14 field; and

15 an advertisement site tracker operative to  
16 track at least one advertisement site detected by the  
17 advertisement site detector without tracking an entire  
18 field.

19  
20 6. Apparatus according to claim 4 or claim 5  
21 wherein said advertisement site detector is operative  
22 to detect at least one advertisement site in at least  
23 one field on a basis other than location of the  
24 advertisement site relative to the sports facility.

25  
26 7. A method for broadcasting advertisements  
27 comprising:

28 broadcasting an image of an event including  
29 at least one advertisement image;

30 identifying the advertisement image within  
31 the image of the event and enhancing only the  
32 advertisement image.

33

34 8. A method according to claim 7 wherein the  
35 step of enhancing comprises the step of sharpening the  
36 edges of the advertisement image.

37

38

1  
2 9. A video processing method comprising:  
3 detecting a moving object in a video sequence  
4 including a plurality of video frames; and  
5 attaching an advertisement to the moving  
6 object in each of the plurality of the video frames.

7  
8  
9 10. Apparatus for advertisement image detection  
10 comprising:  
11 a field grabber operative to grab and  
12 digitize at least one field representing at least a  
13 portion of a sports facility;  
14 an advertisement image detector operative to  
15 detect at least one advertisement image in at least one  
16 field; and  
17 an advertisement exposure time counter  
18 operative to count the length of the exposure time  
19 period of each advertisement image.

20  
21 11. Apparatus according to claim 10 wherein said  
22 time counter is also operative to store at least one  
23 characteristic of the exposure time period of each  
24 advertisement image other than its length, the  
25 apparatus also comprising an advertisement fee computer  
26 operative to compute an advertisement fee according to  
27 the length of the exposure time period and at least one  
28 other characteristic of the exposure time period.

29  
30 12. Apparatus according to claim 11 wherein the  
31 characteristic of the exposure time period other than  
32 the length thereof includes an indication of whether or  
33 not the exposure time period took place during  
34 overtime.

35  
36 13. Apparatus according to claim 11 wherein the  
37 characteristic of the exposure time period other than  
38 the length thereof includes an indication of the

1 interval between each exposure and the beginning of the  
2 game.

3

4 14. Apparatus according to claim 11 wherein the  
5 characteristic of the exposure time period other than  
6 the length thereof includes an indication of temporal  
7 proximity to a significant event.

8

9 15. Apparatus for incorporation of an audio  
10 advertisement into an audio channel representing a  
11 sports event, the apparatus comprising:

12 an audio advertisement memory operative to  
13 store an audio advertisement; and

14 an audio mixer operative to mix the audio  
15 advertisement into an audio channel representing the  
16 sports event.

17

18 16. Apparatus according to claim 15 and also  
19 comprising:

20 a field grabber operative to grab and  
21 digitize at least one field representing at least a  
22 portion of the sports facility at which the sports  
23 event is taking place; and

24 a visual cue detector operative to detect a  
25 visual cue in at least one field and to control  
26 operation of the audio mixer in accordance with the  
27 visual cue.

28

29 17. Apparatus according to claim 16 wherein the  
30 visual cue comprises an advertisement image  
31 corresponding to the audio advertisement.

32

33 18. Apparatus for advertisement site detection  
34 comprising:

35 an advertisement image memory storing an  
36 image of an advertisement to be replaced;

37 a field grabber operative to grab and  
38 digitize at least one field representing at least a

1 portion of a sports facility; and  
2 a field-memory matcher operative to match at  
3 least a portion of a field to at least a portion of the  
4 stored image of the advertisement to be replaced,  
5 thereby to identify an image of at least a portion of  
6 the advertisement to be replaced within the field.

7  
8 19. Apparatus according to claim 18 and also  
9 comprising an advertisement site detector operative to  
10 identify at least one edge of at least one  
11 advertisement site in at least one field.

12  
13 20. A broadcasting method comprising:  
14 imaging an event using a plurality of TV  
15 cameras; and  
16 broadcasting the images generated by the  
17 plurality of TV cameras.

18  
19 21. A method according to claim 20 and wherein  
20 the step of broadcasting also comprises, for each  
21 frame, the step of compressing the images generated by  
22 all but one of the cameras and mixing the compressed  
23 images onto the signal representing the image generated  
24 by the single remaining camera.

25  
26 22. A method according to claim 20 or claim 21  
27 and also comprising the step of receiving the broadcast  
28 at a remote location and deriving therefrom information  
29 regarding at least one advertisement displayed at the  
30 event.

31  
32  
33 23. A method for advertisement site detection  
34 comprising:  
35 grabbing and digitizing at least one field  
36 representing at least a portion of a sports facility;  
37 and  
38 detecting at least one advertisement site in

1 at least one field on a basis other than location of...  
2 the advertisement site relative to the sports facility.

3  
4 24. A method for advertisement incorporation  
5 comprising:

6 grabbing and digitizing at least one field  
7 representing at least a portion of a sports facility;  
8 and  
9 incorporating a portion of an advertisement  
10 into at least one field at a partially occluded  
11 advertisement site within the sports facility, wherein  
12 the portion of the incorporated advertisement  
13 corresponds to an unoccluded portion of the adver-  
14 tisement site.

15  
16 25. A method for advertisement site detection  
17 comprising:

18  
19 grabbing and digitizing at least one field  
20 representing at least a portion of a sports facility;  
21 detecting at least one advertisement site in  
22 at least one field; and  
23 tracking at least one advertisement site  
24 detected by the advertisement site detector without  
25 tracking an entire field.

26  
27  
28 26. A method according to claim 22 wherein the  
29 information regarding the advertisement includes  
30 information pertaining to occlusion of the  
31 advertisement.

32  
33  
34 27. A method for advertisement image detection  
35 comprising:

36  
37 grabbing and digitizing at least one field  
38 representing at least a portion of a sports facility;

1  
2 detecting at least one advertisement image in  
3 at least one field; and  
4 counting the length of the exposure time  
5 period of each advertisement image.  
6  
7  
8

9 28. A method for incorporation of an audio  
10 advertisement into an audio channel representing a  
11 sports event, the apparatus comprising:

12 storing an audio advertisement; and  
13 mixing the audio advertisement into an audio  
14 channel representing the sports event.  
15

16 29. A method for advertisement site detection  
17 comprising:

18  
19 storing an image of an advertisement to be  
20 replaced;

21 grabbing and digitizing at least one field  
22 representing at least a portion of a sports facility;  
23 and

24 matching at least a portion of a field to the  
25 stored image of the advertisement to be replaced,  
26 thereby to identify an image of the advertisement to be  
27 replaced within the field.  
28

29 30. Apparatus according to claim 1 wherein the  
30 advertisement site detector is operative to detect at  
31 least one partially occluded advertisement site in at  
32 least one field on a basis other than location of the  
33 advertisement site relative to the sports facility.  
34

35 31. Apparatus according to claim 4 wherein the  
36 advertisement site tracker is operative to track at  
37 least one partially occluded advertisement site  
38 detected by the advertisement site detector without

1 tracking an entire field.

2

3 32. Apparatus according to claim 18 wherein the  
4 advertisement is partially occluded within the field  
5 and wherein the field-memory matcher is operative to  
6 match only a portion of a field to only a portion of  
7 the stored image of the advertisement to be replaced,  
8 thereby to identify an image of only a portion of the  
9 advertisement to be replaced within the field.

10

11 33. A TV camera array monitoring system for  
12 monitoring a TV camera array comprising at least one TV  
13 camera generating images to be broadcasted, the system  
14 comprising:

15 for each TV camera, a camera FOV center  
16 direction measurement unit;

17 for each TV camera, a lens zoom state  
18 monitoring unit; and a video mixer operative  
19 to mix the outputs of the FOV direction measurement and  
20 lens zoom state monitoring units onto a signal to be  
21 broadcasted.

22

23 34. A system according to claim 33 wherein the  
24 array comprises a plurality of TV cameras and also  
25 comprising an on-air camera identification unit  
26 operative to identify the TV camera from among the  
27 plurality of TV cameras which is currently on-air.

28

29 35. A system according to claim 33 or claim 34  
30 and also comprising a remote advertisement detector  
31 operative to receive the broadcast signal and to  
32 extract the output of the video mixer therefrom.

33

34

35

36

37

38

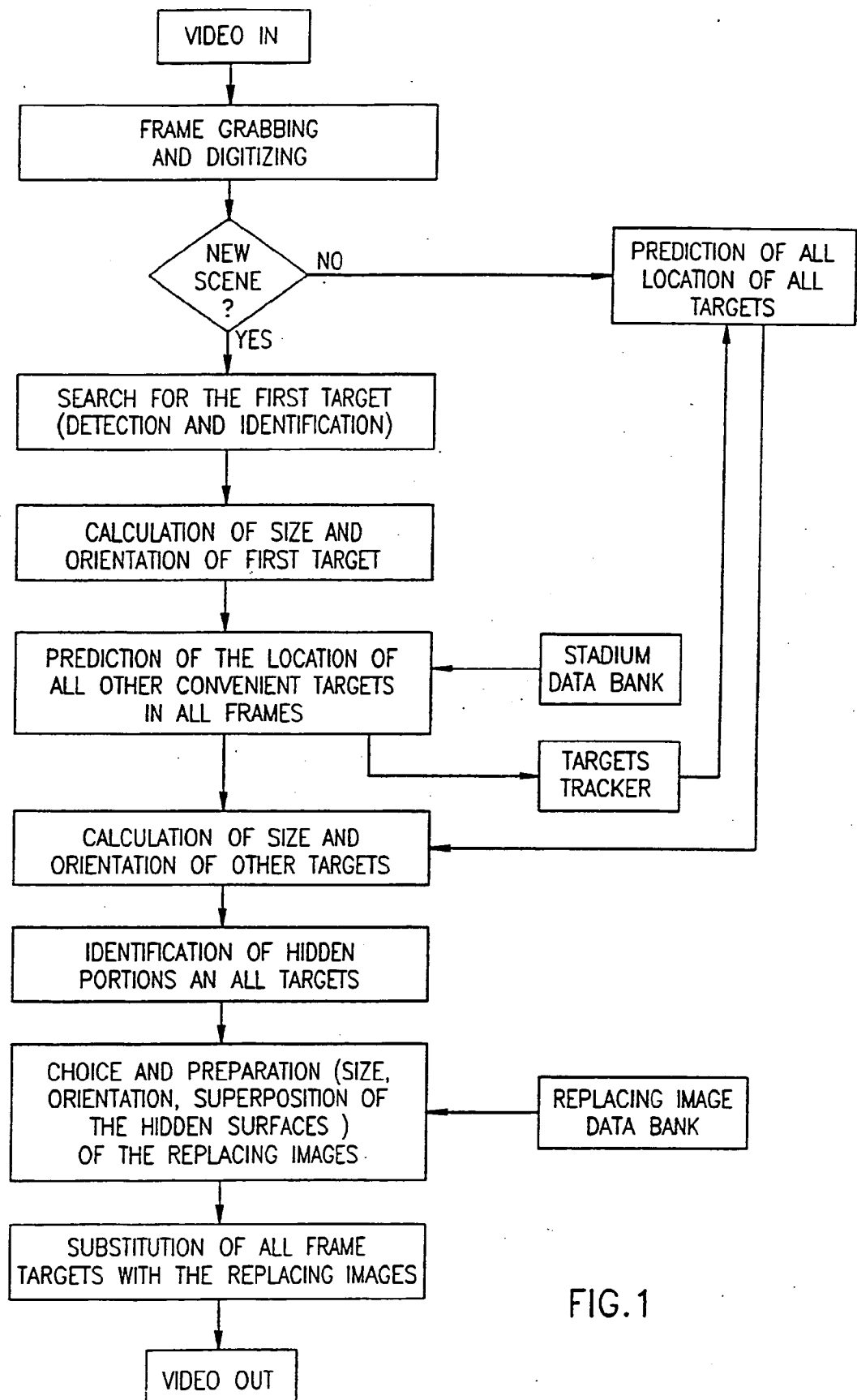


FIG.1



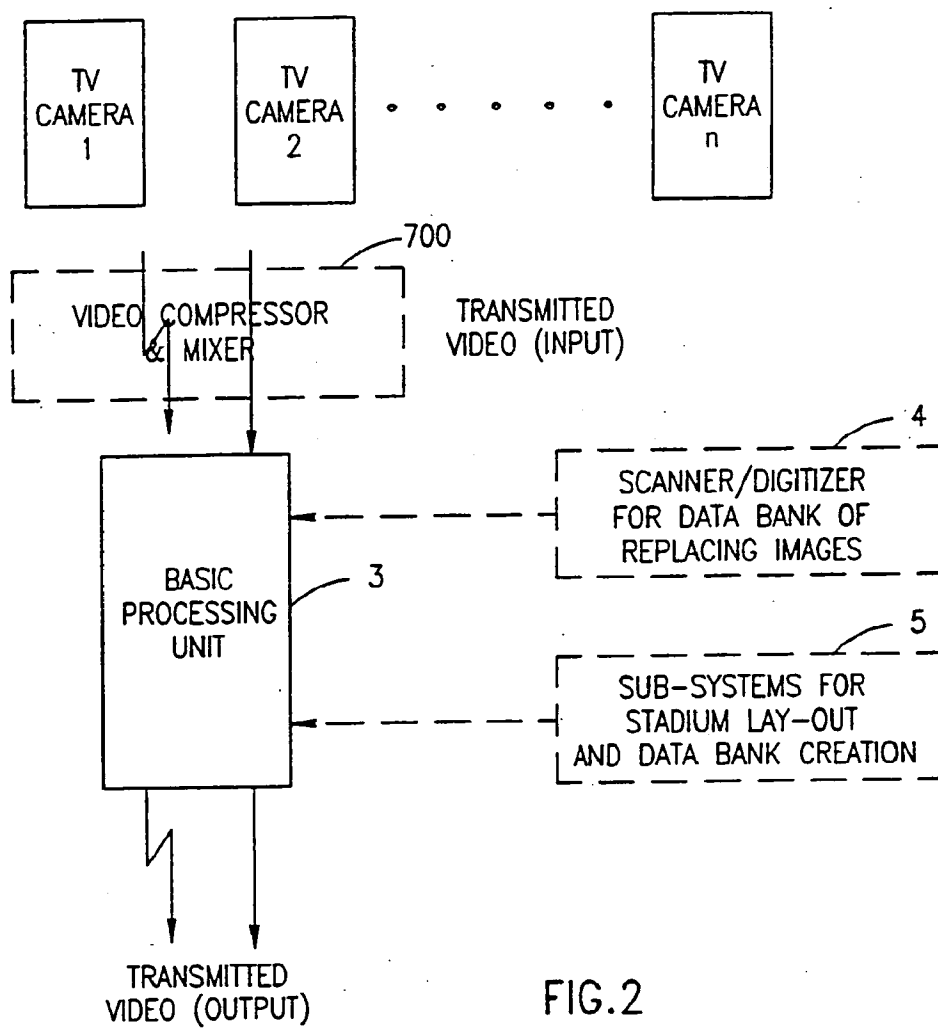


FIG.2

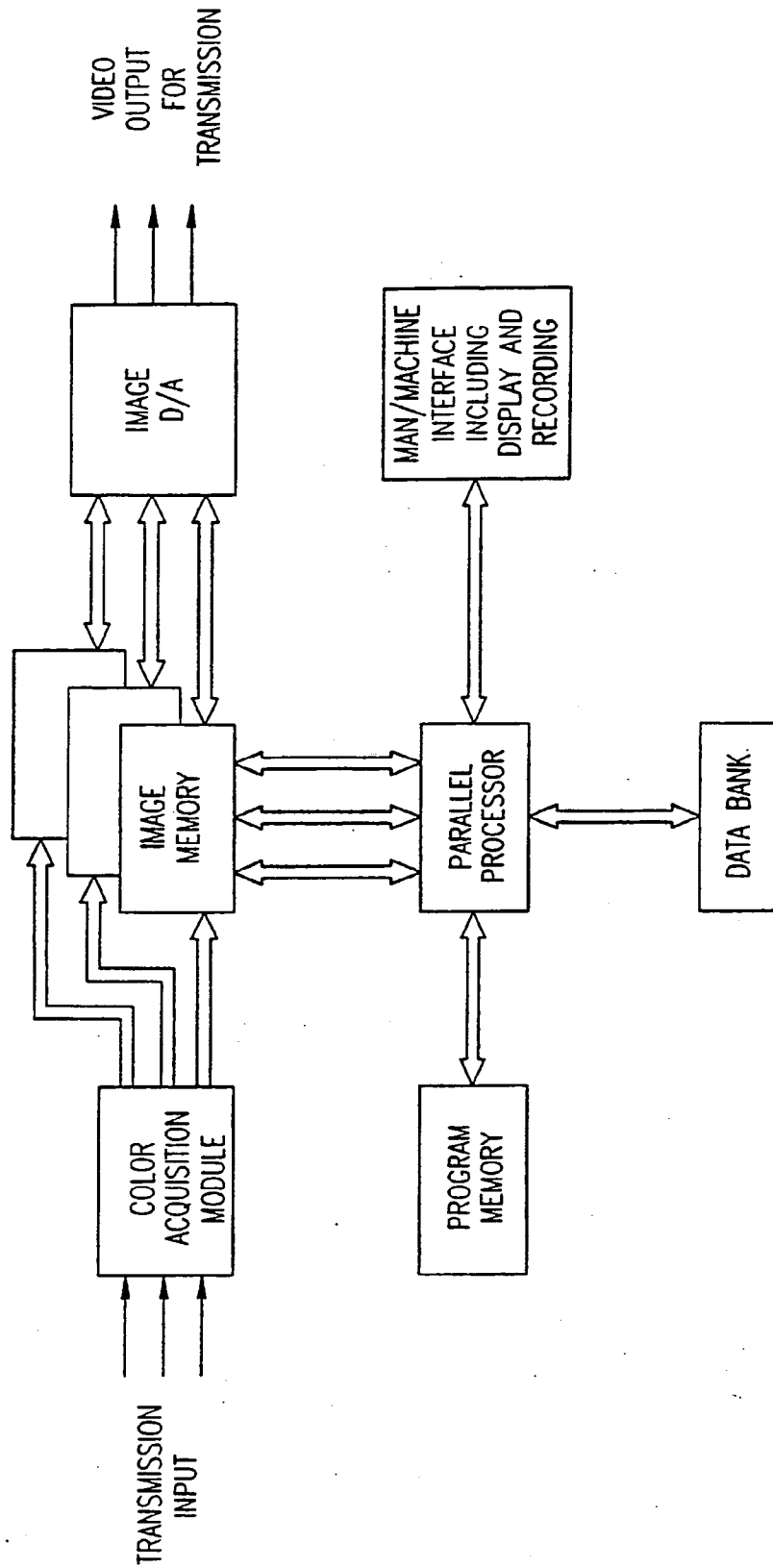


FIG.3

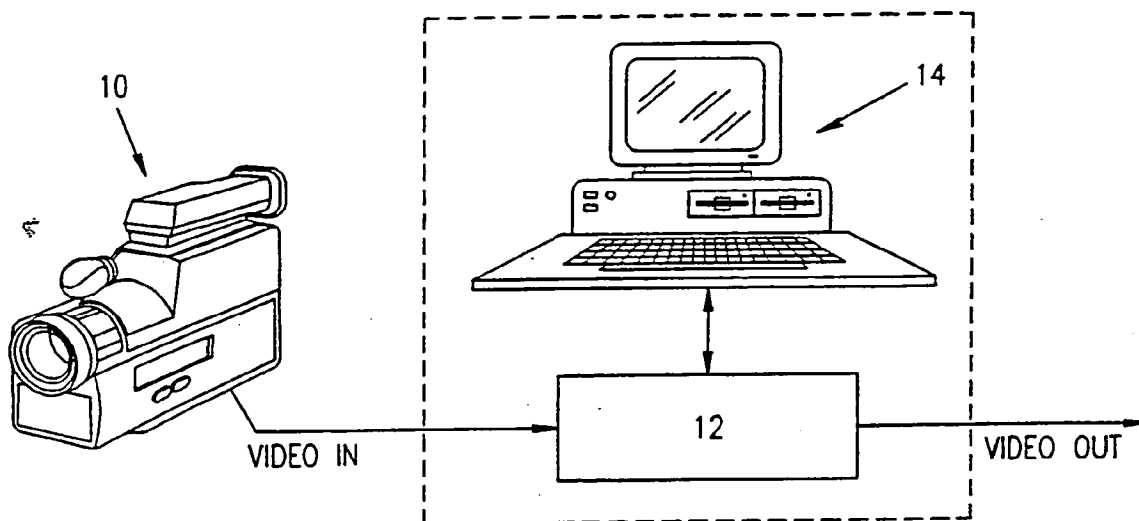


FIG. 4

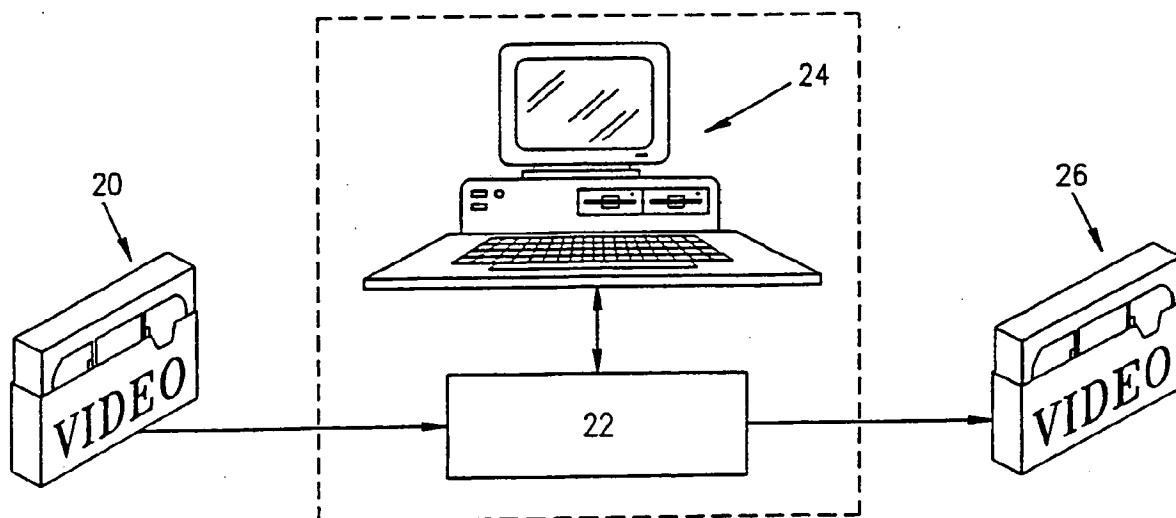


FIG. 5

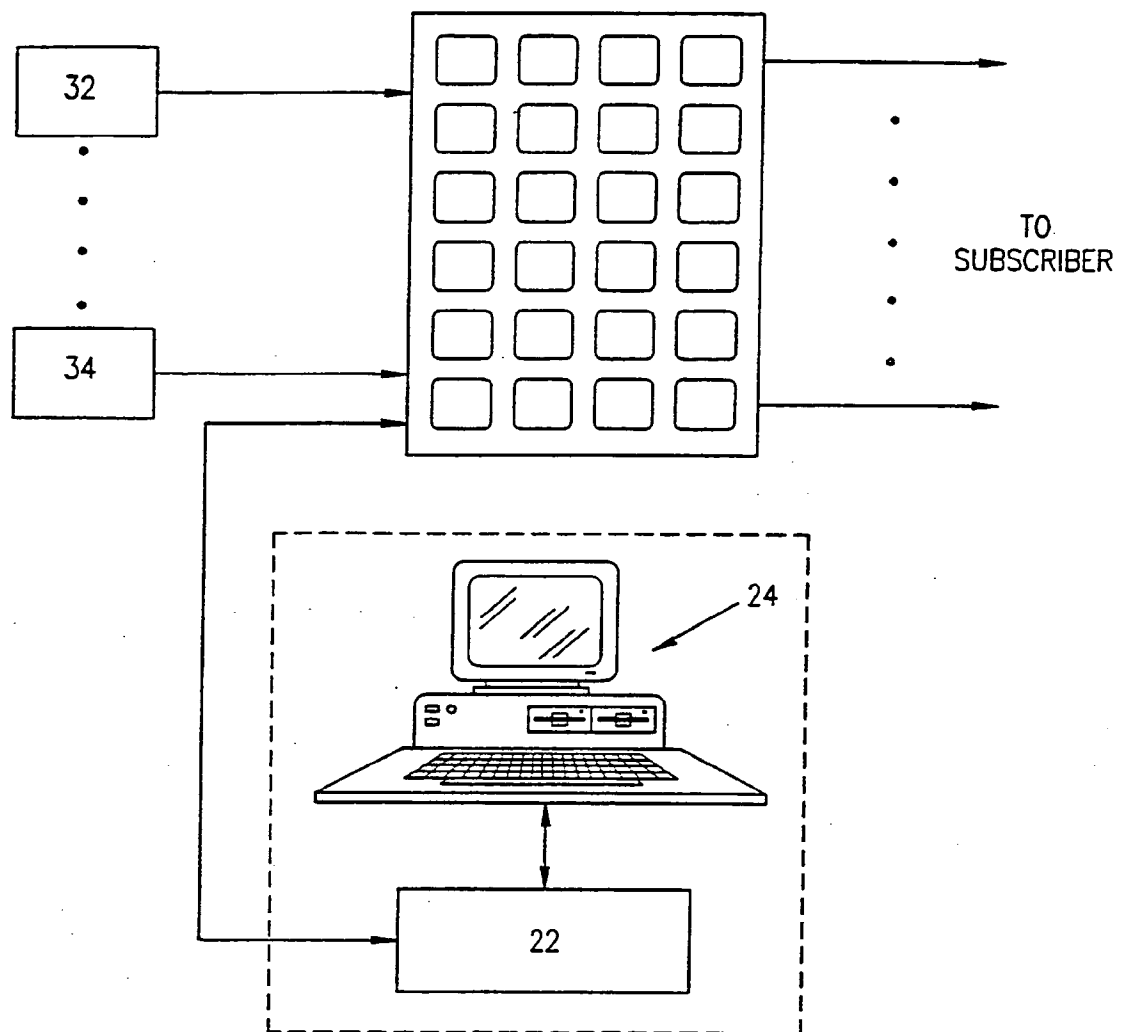


FIG. 6

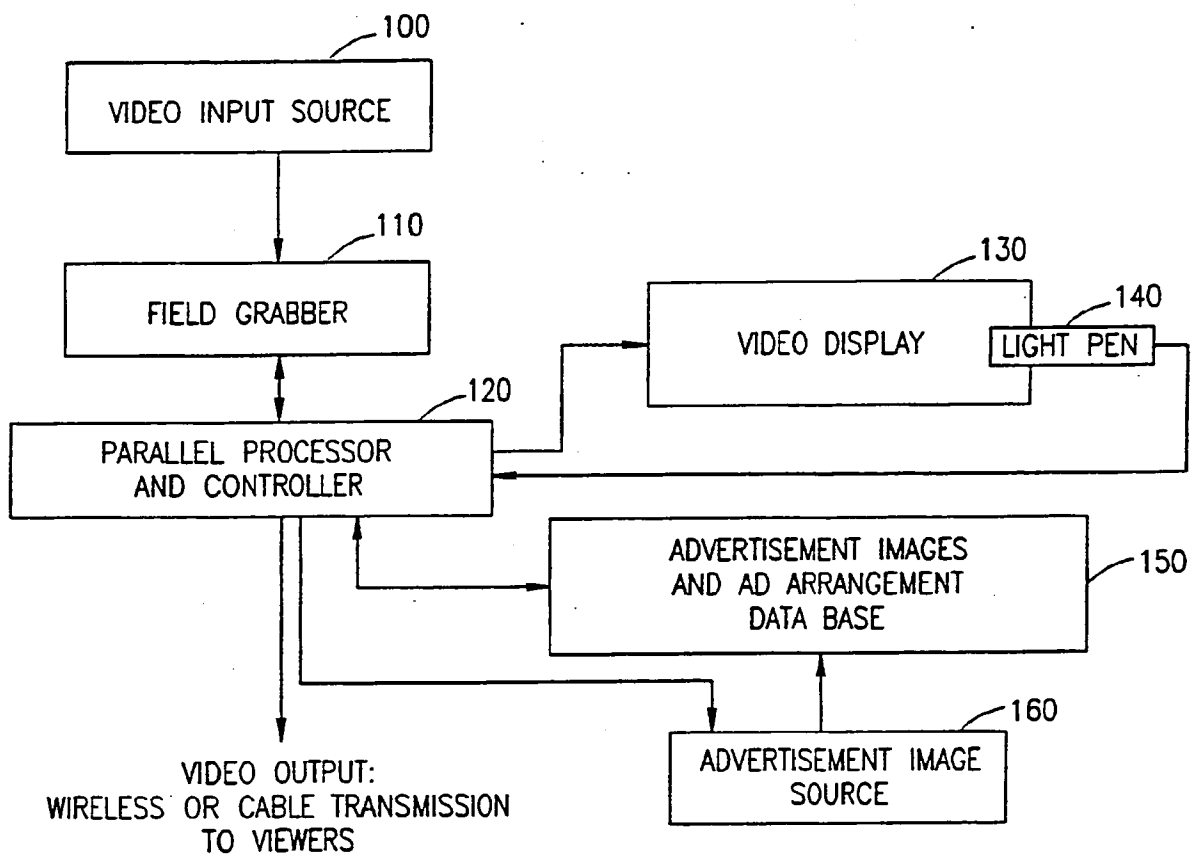


FIG.7

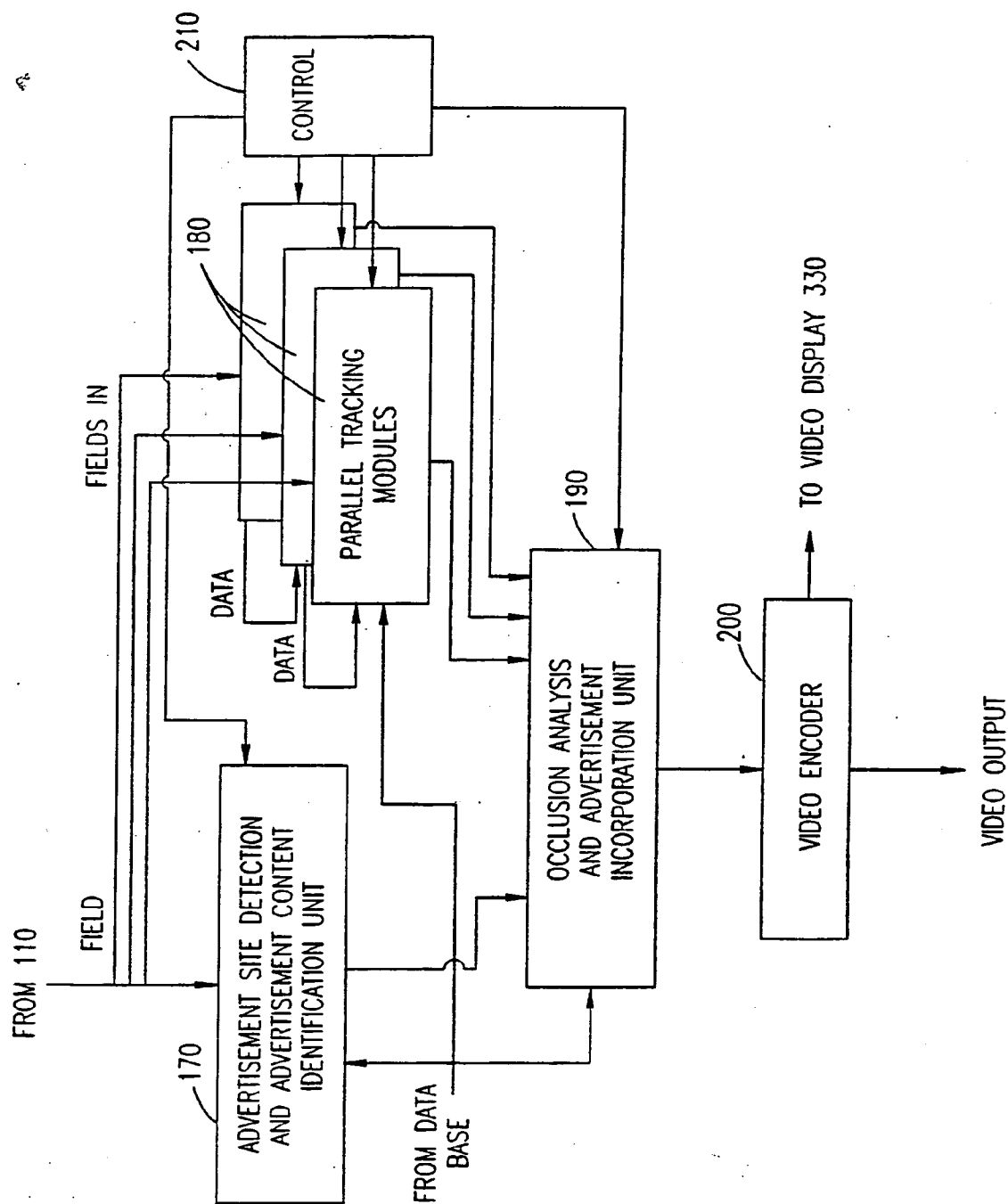


FIG. 8

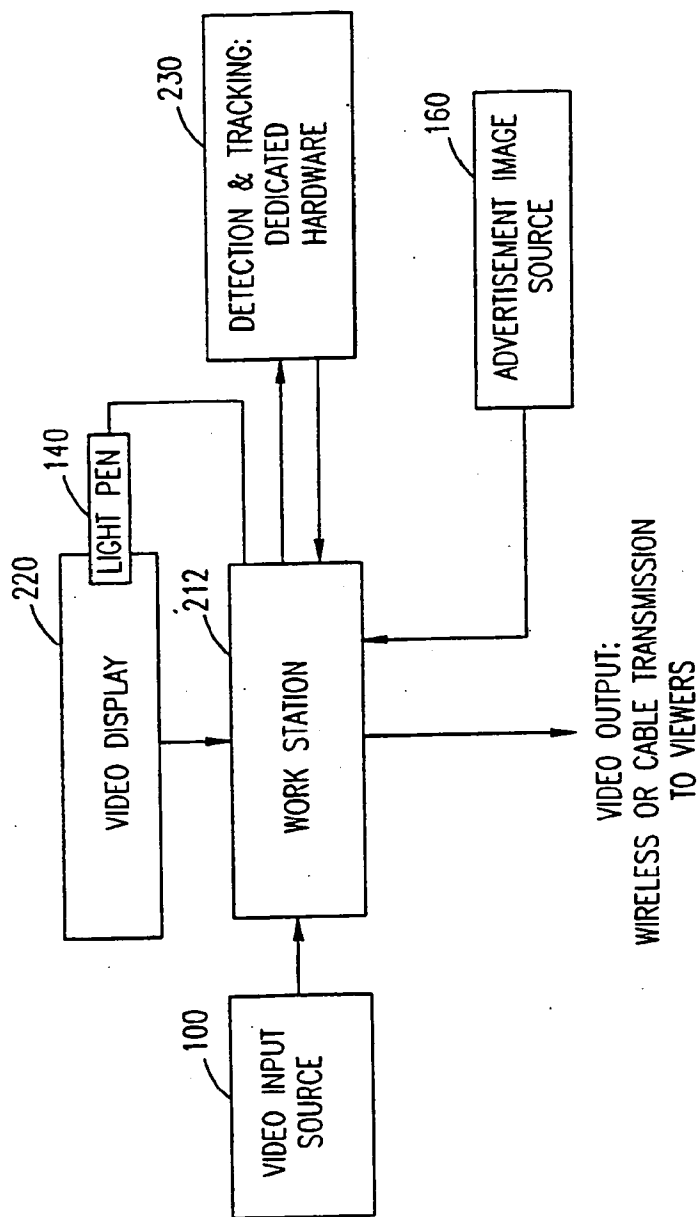


FIG.9

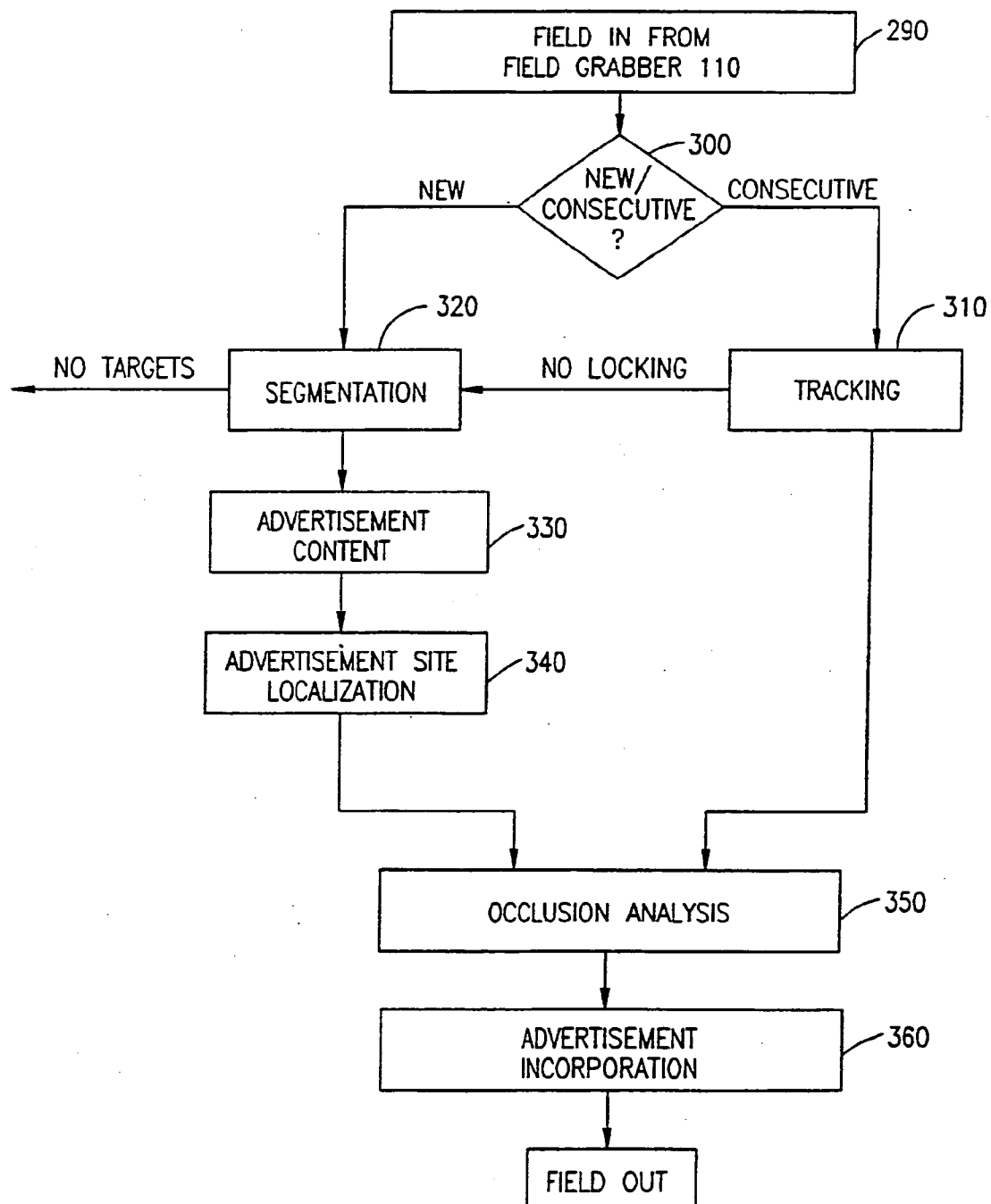


FIG.10A



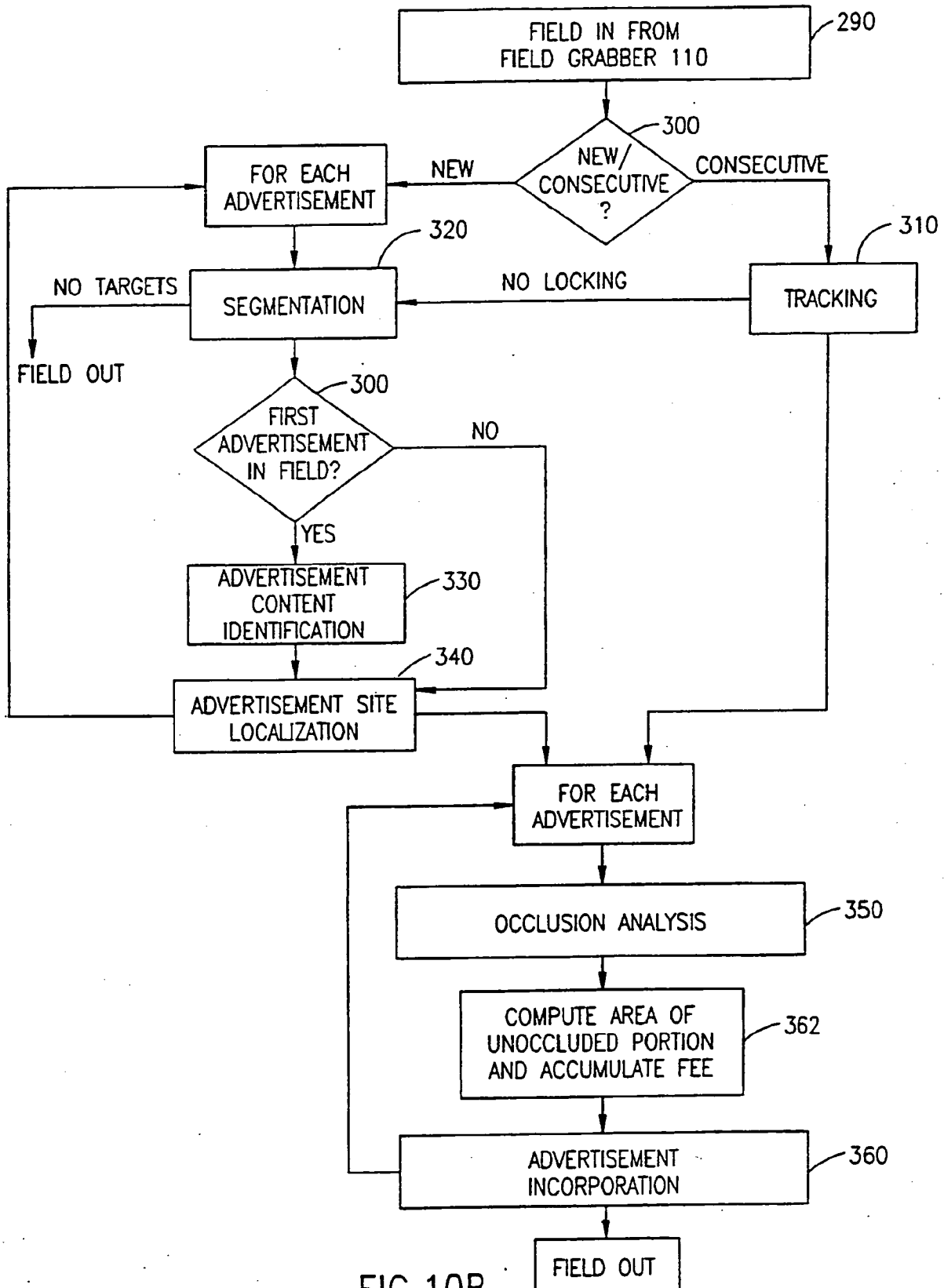


FIG. 10B

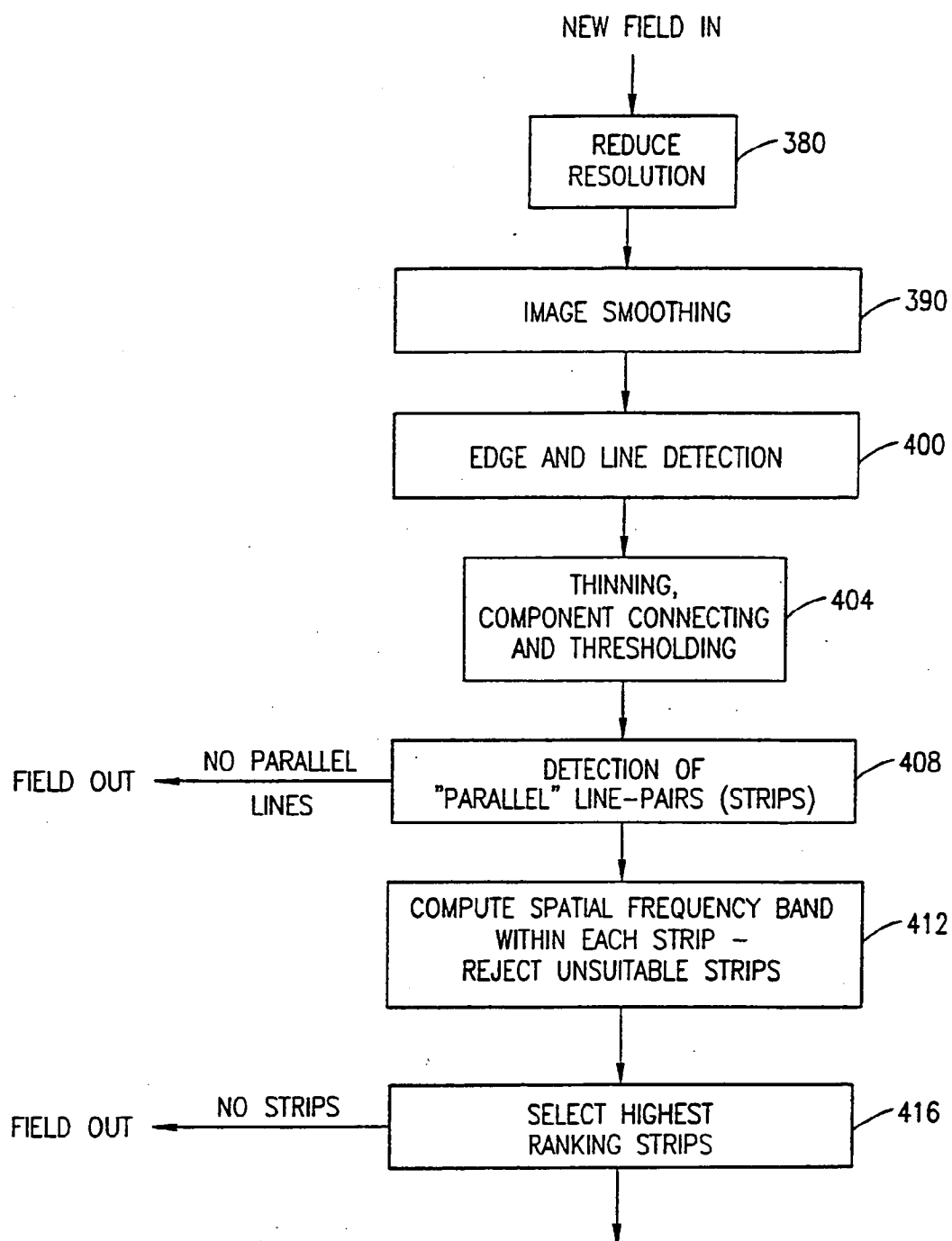


FIG.11

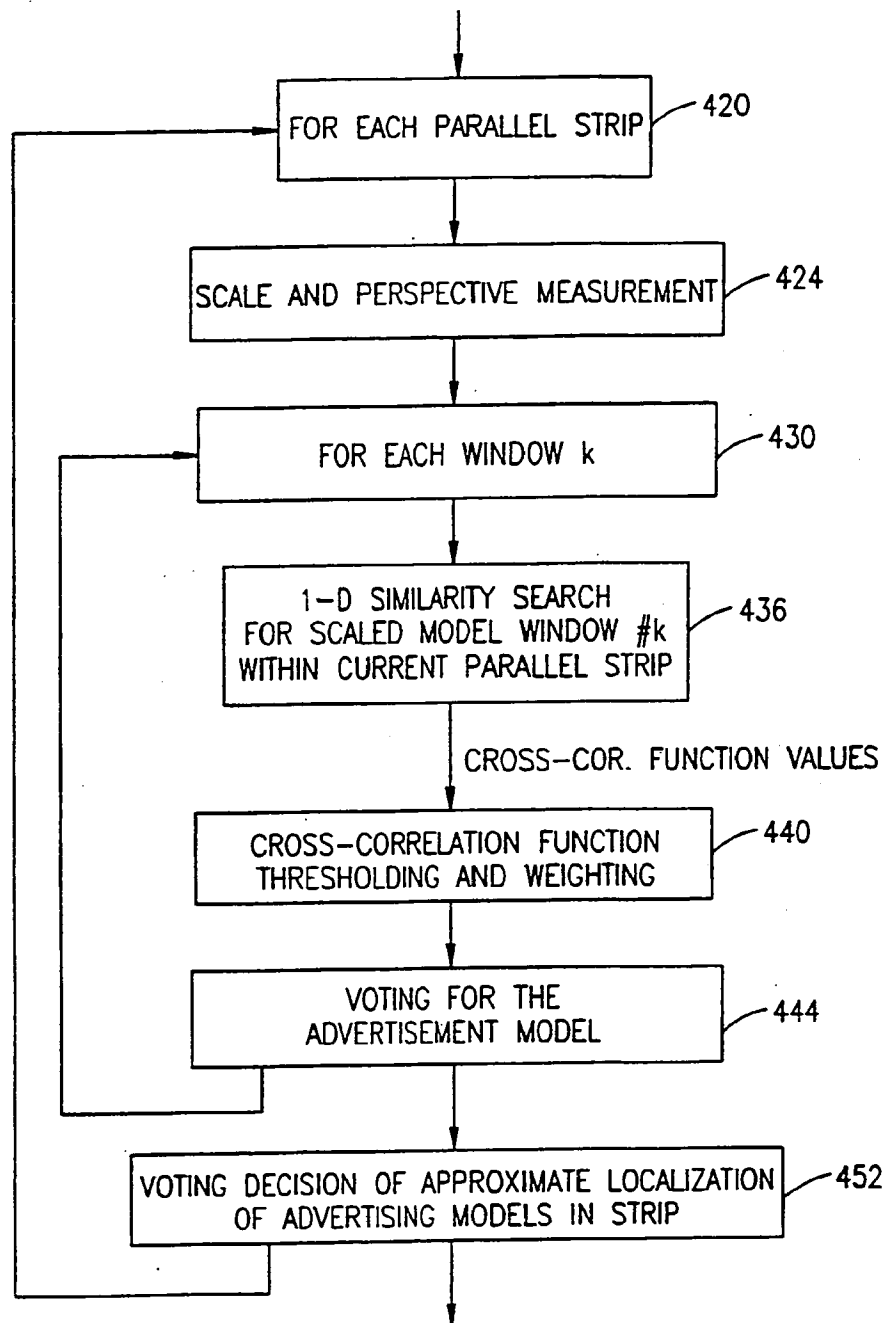


FIG.12

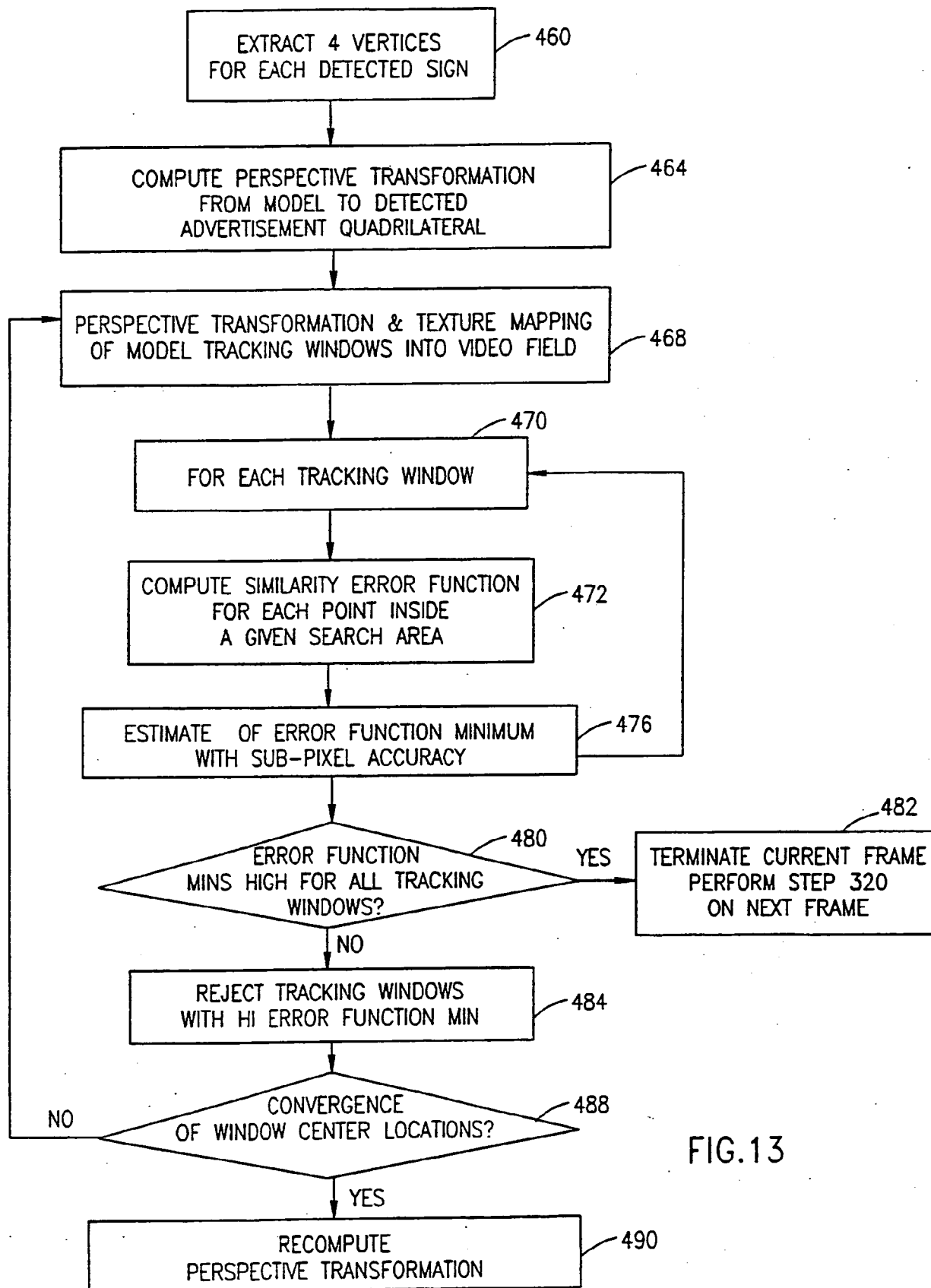


FIG.13

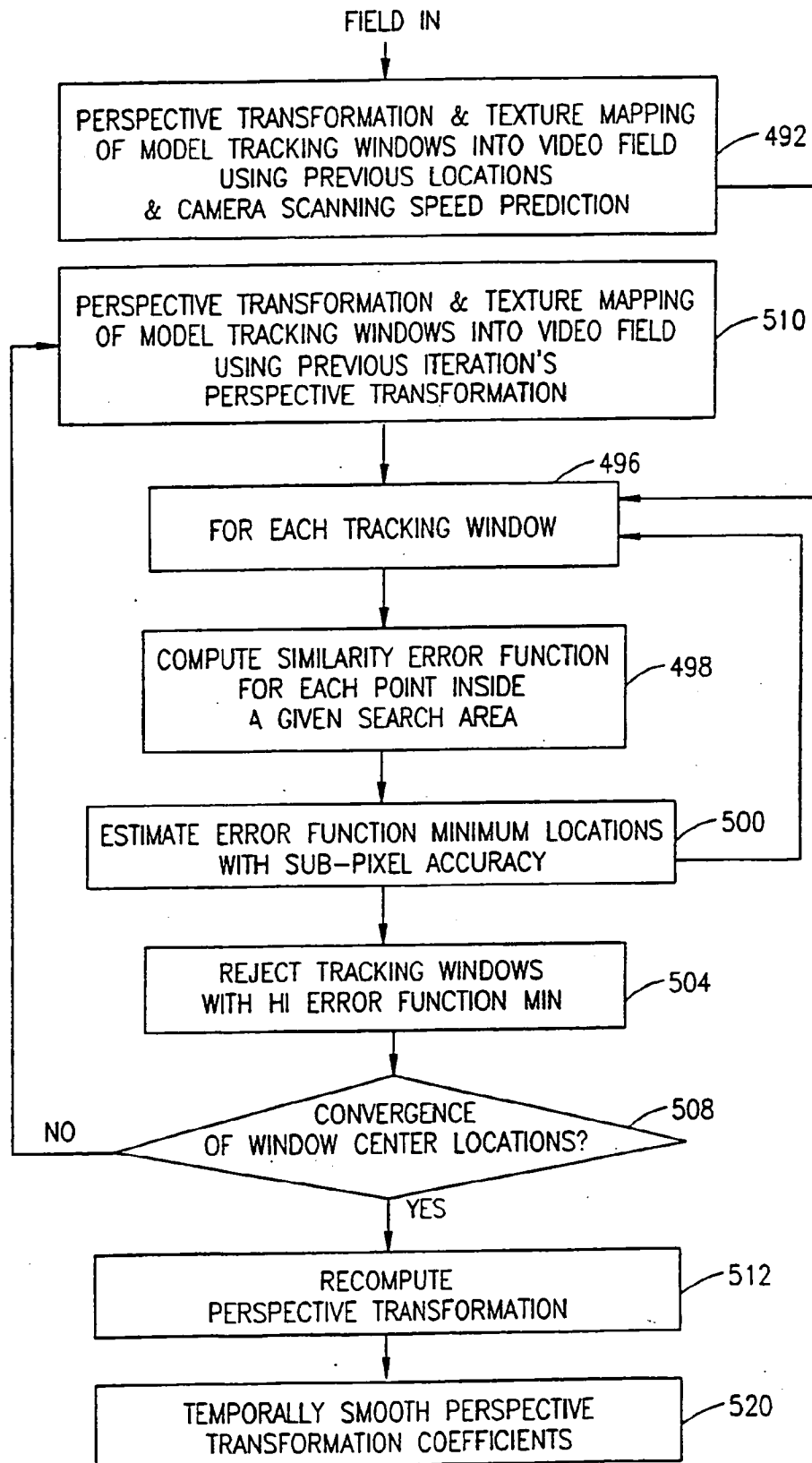


FIG. 14

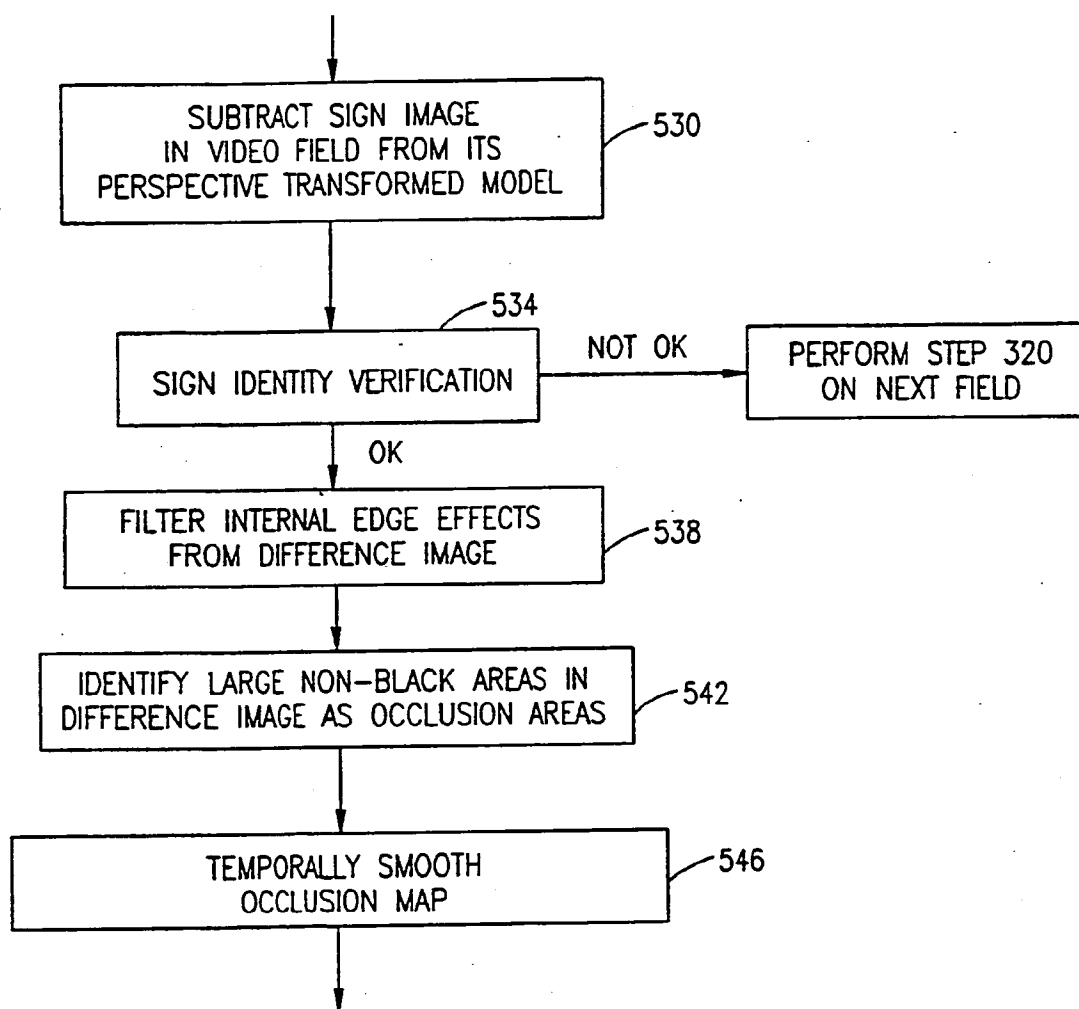


FIG.15

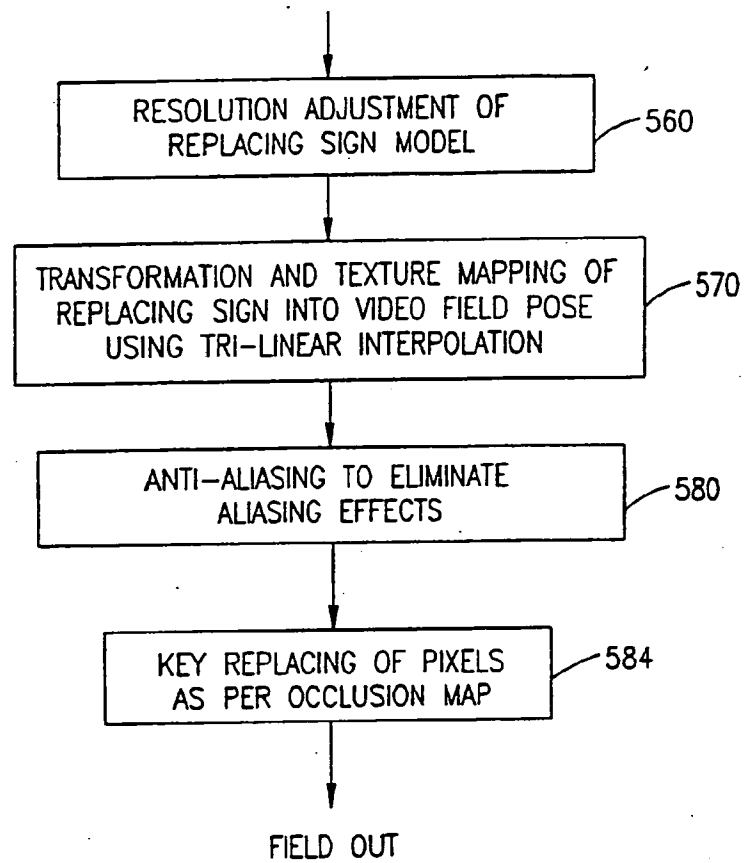


FIG.16

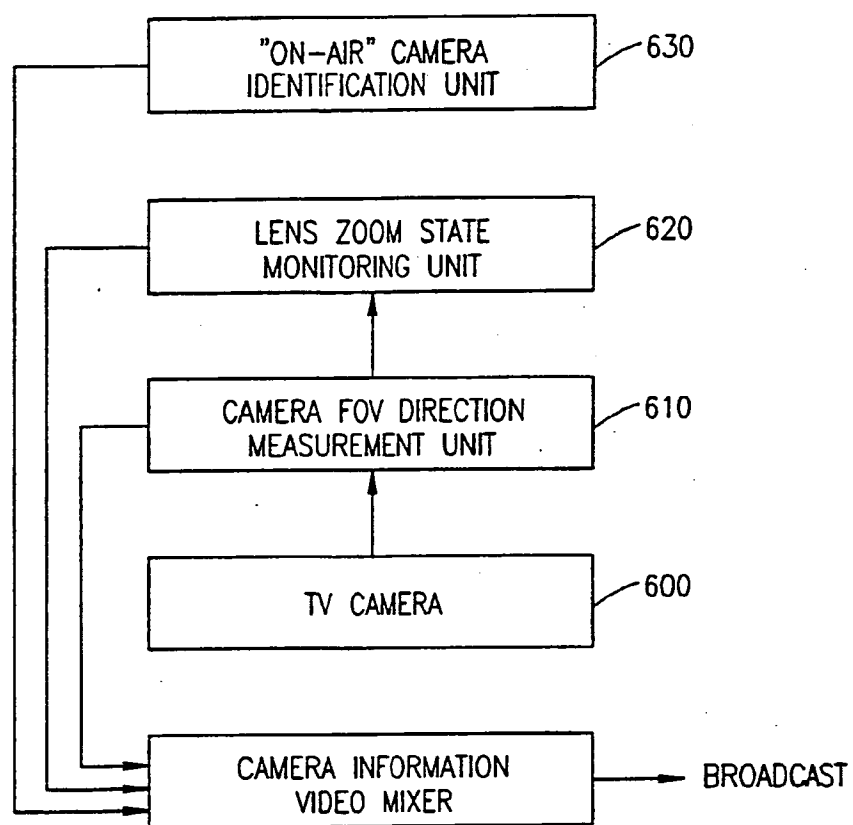


FIG.17



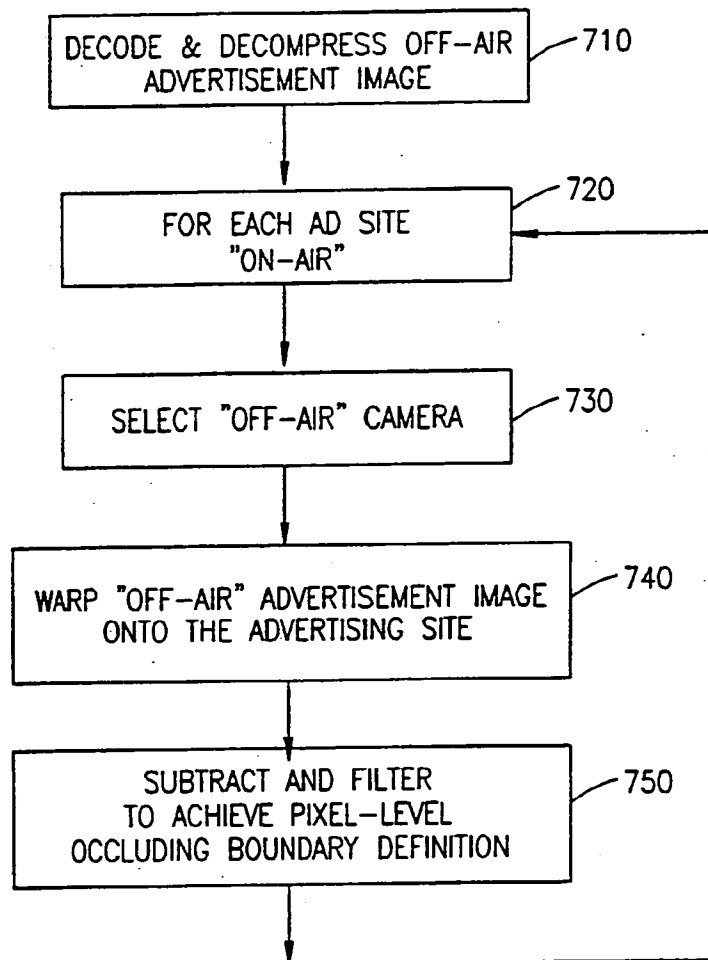


FIG.18

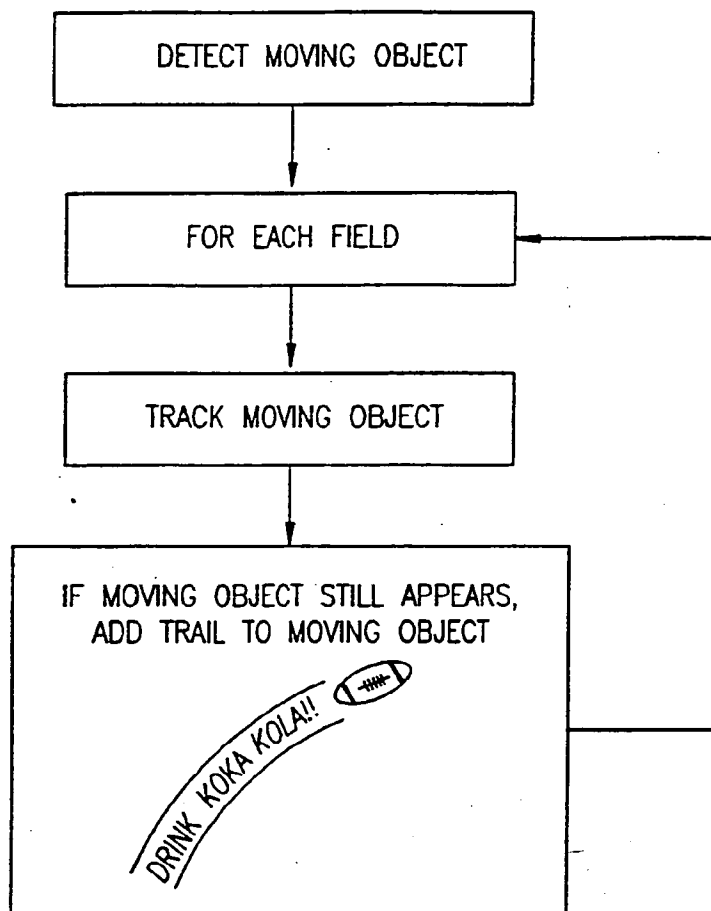


FIG. 19

## INTERNATIONAL SEARCH REPORT

International application No.  
PCT/US94/01679

## A. CLASSIFICATION OF SUBJECT MATTER

IPC(5) :H04N 7/18

US CL :348/138, 169, 463, 559

According to International Patent Classification (IPC) or to both national classification and IPC

## B. FIELDS SEARCHED

Minimum documentation searched (classification system followed by classification symbols)

U.S. : 348/22, 24, 138, 143, 169, 463, 465, 468, 559

Documentation searched other than minimum documentation to the extent that such documents are included in the fields searched  
NONEElectronic data base consulted during the international search (name of data base and, where practicable, search terms used)  
NONE

## C. DOCUMENTS CONSIDERED TO BE RELEVANT

Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
A	US, A, 4,555,726 (TEETER) 26 November 1985, Abstract and Fig. 4.	4, 5, 6/4, 14, 25, 27, 31
A,E	US, A, 5,301,240 (STOCKUM et al) 05 April 1994, Abstract and Fig. 7.	1, 23
A	US, A, 5,021,887 (PARK) 04 June 1991, Abstract and Fig. 5.	1, 23
A	US, A, 5,018,215 (NASR et al) 21 May 1991, Figs. 4 & 5 and col. 5, lines 22-35 & 48 to col. 6, line 2.	1, 4, 23, 25, 31

☐ Further documents are listed in the continuation of Box C. ☐ See patent family annex.

* Special categories of cited documents:	* T	later document published after the international filing date or priority date and not in conflict with the application but cited to understand the principle or theory underlying the invention
* A* document defining the general state of the art which is not considered to be part of particular relevance	* X	document of particular relevance; the claimed invention cannot be considered novel or cannot be considered to involve an inventive step when the document is taken alone
* E* earlier document published on or after the international filing date	* Y	document of particular relevance; the claimed invention cannot be considered to involve an inventive step when the document is combined with one or more other such documents, such combination being obvious to a person skilled in the art
* L* document which may throw doubts on priority claim(s) or which is cited to establish the publication date of another citation or other special reason (as specified)	* Z	document member of the same patent family
* O* document referring to an oral disclosure, use, exhibition or other means		
* P* document published prior to the international filing date but later than the priority date claimed		

Date of the actual completion of the international search

13 JUNE 1994

Date of mailing of the international search report

05 AUG 1994

Name and mailing address of the ISA/US  
Commissioner of Patents and Trademarks  
Box PCT  
Washington, D.C. 20231

Facsimile No. (703) 305-3230

Authorized officer

Jeffrey Murrell

Telephone No. (703) 305-8155

# INTERNATIONAL SEARCH REPORT

International application No.  
PCT/US94/01679

## Box I Observations where certain claims were found unsearchable (Continuation of item 1 of first sheet)

This international report has not been established in respect of certain claims under Article 17(2)(a) for the following reasons:

1. ☐ Claims Nos.:  
because they relate to subject matter not required to be searched by this Authority, namely:
2. ☐ Claims Nos.:  
because they relate to parts of the international application that do not comply with the prescribed requirements to such an extent that no meaningful international search can be carried out, specifically:
3. ☐ Claims Nos.:  
because they are dependent claims and are not drafted in accordance with the second and third sentences of Rule 6.4(a).

## Box II Observations where unity of invention is lacking (Continuation of item 2 of first sheet)

This International Searching Authority found multiple inventions in this international application, as follows:

Please See Extra Sheet.

1. ☐ As all required additional search fees were timely paid by the applicant, this international search report covers all searchable claims.
2. ☐ As all searchable claims could be searched without effort justifying an additional fee, this Authority did not invite payment of any additional fee.
3. ☐ As only some of the required additional search fees were timely paid by the applicant, this international search report covers only those claims for which fees were paid, specifically claims Nos.:
4. ☒ No required additional search fees were timely paid by the applicant. Consequently, this international search report is restricted to the invention first mentioned in the claims; it is covered by claims Nos.:  
1, 4, 6/4, 10-14, 23, 25, 27, 30-31

Remark on Protest

- ☐ The additional search fees were accompanied by the applicant's protest.  
☐ No protest accompanied the payment of additional search fees.

# INTERNATIONAL SEARCH REPORT

International application No.  
PCT/US94/01679

## BOX II. OBSERVATIONS WHERE UNITY OF INVENTION WAS LACKING

This ISA found multiple inventions as follows:

- I. Claims 1, 4, 10-14, 23, 25, 27, 30-31 are drawn to the detection of sports advertisement information, classified in Class 348, subclasses 465 and 468.
- II. Claims 7-9 and 20-21 are drawn to the transmission and receiving of broadcast advertisement information, classified in Class 348, subclasses 432, 473 and 563.
- III. Claims 2-3, 5, 15-19, 24, 28-29 and 32 are drawn to the incorporation of advertisement information into a field signal, e.g. an audio field signal, classified in Class 348, subclassis 462, 476, 482 and 484.
- IV. Claims 33-35 are drawn to the details of television cameras for monitoring broadcasts, classified in Class 348, subclasses 187 and 192.